Centers for Medicare & Medicaid Services (CMS)

Requirements Writer's Guide
Version 3.1

October 2004

Prepared by:

Office of Information Services
CIO's Planning, Management, and Support Group

and

G E N O V A
T E C H N O L O G I E S

Table of Contents

1 Introduction

Whether building a new system, making changes to existing software or using commercial off-the-shelf software, exploring, capturing and communicating requirements is necessary. Requirements are a universal language commonly used to communicate ideas within most organizations. They are usually derived from a variety of informational sources and can represent many things to many people. Because requirements can represent many things, defining them is a form of art that is commonly used to express the needs and wants of a system in its simplest form.

In the world of systems development, requirements represent the things needed to know and the things needed to do that are critical for project success. The right product cannot be built unless it is known precisely what the product should do and how the product's success is to be measured. Projects succeed when there are clearly defined goals and attainable results, which focus on satisfying the need and delivering results users want. This is only possible when those needs are clearly defined and the problem is fully documented in writing.

The person responsible for writing requirements must understand the business goals and have knowledge of the intended product, what it has to do, what qualities it must have, what constraints it must conform to and what interfaces it must have to the outside world. As development continues, knowledge of the product increases and the analytical activity should continue to grow until all the requirements are known and analyses at all phases are complete. In this way, the entire analysis process deals with identifying the 'problem'.

1.1 Purpose

The purpose of this document is to provide guidance to managers, analysts, users and contractors who are responsible for scoping, writing or reviewing requirements. It seeks to describe the level of detail that should be included in each written requirement and what each engineer must look for when conducting formal reviews of internally or externally developed requirements. It also intends to standardize the nomenclature associated with writing requirements. The document employs an object oriented methodology for organizing requirements.

This document will assist in effectively managing requirements as well as serve as a standard guideline by which all requirements shall be measured and deemed acceptable. Lastly, the document provides a clearly defined structure for written requirements and a standardized set of criteria to assess each requirement.

1.2 Scope

This guide addresses how to write and organize business, functional and nonfunctional requirements. The intended audience for this guide is comprised of CMS business owners, project owners, requirements engineers, project teams and contractors charged with developing business, functional and nonfunctional requirements.

CMS follows the guidance specified in Section 6.26.3 of IEEE/EIA 12207.1 1997 Guide for Information Technology with respect to requirements. The IEEE standard recommends tailoring its methodology to fit the needs of the organization using it. Captured within this Writer's Guide are the results of this tailoring.

This guide provides descriptions of guidelines and templates to assist in the development of requirements. Although it is encouraged that these guidelines be rigorously followed, the unique characteristics of each IT project may warrant slight modifications to these guidelines and/or templates.

1.3 Document Organization

Regardless of whether the readers are novices or experienced requirements writers, it is strongly recommended that they familiarize themselves with the terms listed in the glossary. Novice guide users may find it helpful to refer to the glossary as the terms are used throughout the document. Experienced requirements documenters should review the glossary to determine if their understanding of a term coincides with how the guide intends to use it.

Readers unfamiliar with their role in the requirements writing process should first examine Chapter 2. This chapter identifies the tasks and responsibilities for the various people involved in the requirements writing process. Following this, the reader may refer to the appropriate sections in Chapter 3 that step by step guides them through the process of fulfilling each of the tasks they are responsible for completing. By contrast, if readers are familiar with what is expected of them, they should be able to refer to the Table of Contents and the Index to quickly locate specific, relevant sections.

It is recommended that all readers familiarize themselves with Chapter 4 where stylistic and syntactical writing considerations are discussed. Stylistic considerations are of importance to all writers and syntactical rules are especially important to Requirements Engineers because they standardize communication between the reader, the requirements engineer and the designer, thus providing more clarity.

Appendix B includes a sample Requirements Document (RD) for reference in addition to the samples included in the text. Nonfunctional requirements will be a subsection of this document.

1.4 Source Documentation

IEEE/EIA Guide for Information Technology, Std 12207.1
IEEE Developing System Requirements Specification, Std 1233
IEEE Recommended Practice for Software Requirements Specifications, Std 830
The Art and Discipline of Writing Requirements, Revision 2
Common Working File Redesign (CWFR) Project Technical Writer's Guide, Release 3.7
Business Case Analysis Development Guide, Version 3.0

1.5 Process Hierarchy

This guide is based on a process of building a system that starts at a very broad level (Strategic Goals) and generates more detail about the system in successive levels. The need for a new system usually begins with either legislation from Congress or a strategic initiative from CMS Management. The Business Owner then translates these goals into Business Requirements. The Project Owner uses the Business Requirements to create more detailed Events, Scenarios and Functional and Nonfunctional Requirements with the help of Requirements Engineers and Subject Matter Experts. The Requirements Document is then turned over to the Systems Designer where the process of solving the problem begins.

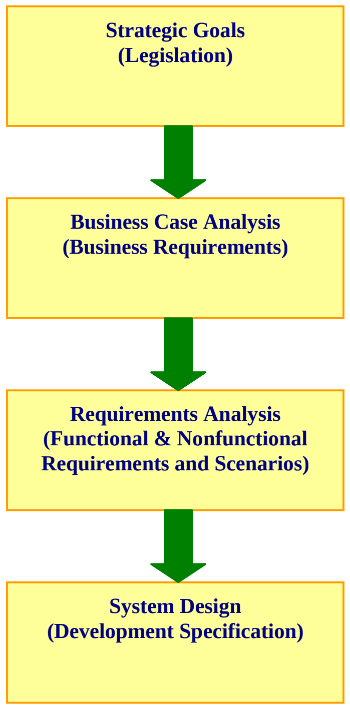


Figure 1 - Process Hierarchy

1.6 Glossary

Term	Process Hierarchy Level (see Section 1.5)	Definition
Alternate Scenario	Requirements Analysis	A scenario identifying the steps that occur when a scenario's outcome is different from expected.
Business Requirements	Business Case Analysis (BCA)	Business requirements are statements of the functions or program needs that must be met in order to accomplish the business objectives of the project. These business objectives address legislative mandates or strategic business goals (such as improved customer service, business efficiencies, business process reengineering, etc.).
Business Rule	Requirements Analysis	The conditions under which a functional requirement must operate. This includes the function's triggers, preconditions and scenarios.
Constraint	Business Case Analysis & Requirements Analysis	A specific type of requirement that limits design, development, or deployment options. Constraints mandate the way the system must be produced and may exist at both the Business Requirements level and the Functional/Nonfunctional Requirements level.
Domain	Business Case Analysis	A user or system that interacts with the system being designed (actor).
Event	Requirements Analysis	A business process that occurs between the system and a domain or a process internally initiated by the system.
Functional Requirement	Requirements Analysis	An action or expectation of what the system will take or do. It is measured by concrete means like data values, decision making logic and algorithms.
Nonfunctional Requirement	Requirements Analysis	This is a behavioral property the eventual system must have. See Section 3.3.2.1 for a list of various types.
Pass/Fail Statement	Requirements Analysis	A testable statement that describes how to know if a requirement was met.
Precondition	Requirements Analysis	The state the system must be in before a scenario can start.
Requirement	All levels	A measurable statement of intent about something that the system must do, a property the system must have or a constraint on the system.
Scenario	Requirements Analysis	A sequence of steps taken to complete an event.
Stakeholder	Business Case Analysis	A person who interacts with, is responsible for or has some interest in the system being developed.

System Requirement	Requirements Analysis	An umbrella term, which includes functional and nonfunctional requirements, to define what a system must do in order to satisfy the Business Requirements. As such, system requirements are lower level requirements that address the specific characteristics that the system must address in order to be acceptable as an end product.
Traceability	Requirements Analysis	The ability to trace relationships between two or more requirements.
Trigger	Requirements Analysis	An action or activity that occurs that causes the scenario to begin.
User Requirement	Requirements Analysis	A specific type of functional or nonfunctional requirement that relates to human interaction with the system.

Please see the Common Working File Redesign terms in Appendix D to understand how these terms are used differently in some redesigned systems.

2 Guide Users

There are four types of users this guide is intended to help:

User Types	Icon
Business Owners	💼
Project Owners	📊
Requirements Engineers	📐
Subject Matter Experts	🎓

The guide will help each type of user identify what is expected of them throughout the requirements documentation process. The icon associated with a user type will identify sections that apply to the user type.

2.1 For the Business Owner

The Business Owner(s)/Partner(s) are the entity or entities responsible for defining, promoting, endorsing and upholding the business needs and user requirements for the system and for performing user acceptance testing of the final product(s) based on those business needs and requirements. The Business Owner(s)/Partner(s) may represent the interests of external partners such as Medicare Contractors, Fiscal Intermediaries or other Federal agencies and/or the interests of internal CMS Centers, Offices or Consortia. The Business Owner(s)/Partner(s) define and verify system functionality, access rights, business rules and the privacy classification, timeliness, completeness and accuracy of data. The Business Owner may also be the Project Owner.

Below are the tasks for which the business owner is responsible in the requirements writing process. If a Business Case Analysis (BCA) was written for the project, the Business Owner should be thoroughly familiar with it. This will be of great assistance and in some cases will provide the information necessary to perform the following tasks.

1. Writing a Functional Purpose
2. Writing a Business Purpose
3. Determining the Measures of Success
4. Assessing the Mission Profile
5. Identifying the Stakeholders
6. Determining the Risks & Assumptions
7. Writing the Business Requirements

Sections 3.1.1 – 3.1.7 detail these tasks. Appendix B provides samples for each task.

2.2 For the Project Owner

The Project Owner/Manager is the individual at CMS who has the day-to-day responsibility for the success and management of the IT project. This individual may be at any level in the CMS organizational structure. The Project Owner/Manager is the primary point of contact for the project with relation to the IT Investment Management Process. The Project Owner/Manager communicates regularly with management on the needs and progress of the project and is responsible for obtaining and managing project resources. In those cases where contract support is utilized, the Project Owner/Manager may also serve as the Government Task Leader (GTL) and/or the Project Officer, or otherwise provide support to the assigned GTL and Project Officer for the contract. The Project Owner may also operate as a Requirements Engineer.

2.2.1 Scoping Requirements

It is essential that the Project Owner have the business strategy documentation produced by the Business Owner in order to begin. The Work Context Diagram (see Section 3.2.1 below) is the first task to complete. In some cases, the BCA will include a preliminary Work Context Diagram and it is the responsibility of the Project Owner to ensure it is complete.

If the Project Owner is also operating as part of the Requirements Engineering Team, they should refer to the next section as well as Section 3.3 in order to start the requirements writing process.

2.2.2 Managing Requirements

The Project Owner is responsible for managing the requirements during requirements development as well as after the team has finished its work. This includes ensuring requirement traceability, reviewing requirements, baselining requirements (both formal and informal), tracking change requests (when appropriate) and enforcing writing standards. Sections 3.2.3 – 3.2.6 describe these tasks.

It is also the Project Owner's responsibility to manage the Requirements Document. The parts of this document are discussed throughout the guide and a sample document is included in Appendix B.

2.3 For the Requirements Engineer

The Requirements Engineer is an individual tasked with gathering and writing requirements for the IT project. The person may also be the Project Owner, an employee of the CMS organization or a contractor. The Requirements Engineer communicates regularly with the project team and the Subject Matter Experts while gathering and writing the requirements.

After performing the requirements gathering process, the Requirements Engineer is responsible for putting the requirements into a formal structure that may be handed over to system designers to continue the project. The steps an engineer must take to accomplish this are as follows:

1. Documenting Events (if not provided with the Work Context Diagram)
2. Creating Scenarios
3. Writing Functional Requirements
4. Writing Nonfunctional Requirements
5. Determining Pass/Fail Statements
6. Importing Requirements into DOORS

These tasks are described in Section 3.3. While not responsible for conducting the review of requirements, the Requirements Engineer should be familiar with the process as described in Section 3.2.4.

2.4 For the Subject Matter Expert

A Subject Matter Expert (SME) is an employee of the CMS organization or a contractor with specific knowledge regarding a domain that interacts with the system. A SME should always be a stakeholder in the project, though not all stakeholders are SMEs.

While a SME is not responsible for writing requirements or the supporting documentation, they should be familiar with what the project team will expect from them during this process so that the team's deliverables will accurately reflect the SME's needs and knowledge.

SMEs should be involved throughout the entire process of requirements documentation and should specifically be familiar with:

1. Chapter 3.1: Project Management tasks
2. Chapter 3.2.1: Work Context Diagram task
3. Chapter 3.3.1 – 3.3.3: Documenting Requirements tasks
4. Chapter 3.2.4: Reviewing Requirements task

3 Guide User Tasks

This section describes specific tasks and deliverables to complete as determined by the reader's role in the project. See Chapter 2 if you are unsure of which tasks require your participation.

3.1 Business Strategy

All the stakeholders that the BCA or project kickoff identified should participate in performing the Business Strategy tasks. Project Owners cannot begin their work until these tasks are complete. The information collected in these tasks make up part of the Functional Requirements Document.

3.1.1 Writing a Functional Purpose

The functional purpose describes *what* the project shall do for the business in a single, complete sentence. It is not unusual for stakeholders to have competing notions regarding what the functional purpose should be. While a compound sentence is acceptable, the purpose should still be limited to one sentence.

The purpose should be a statement containing strong transitive verbs like: build, compare, explore, investigate, perform, recommend or replace. This is the statement that gives the entire project its direction. When formulating the statement, consider the seriousness of the problem and why it needs to be solved.

For example, the project team for the Undocumented Alien Reimbursement System (UARS) Project came up with:

> The functional purpose is to build an enrollment and payment processing system to reimburse health care providers (hospitals, physicians, ambulance providers and Indian Health and tribal organizations) for the emergency treatment of undocumented aliens.

If a functional purpose cannot be reduced to a single sentence, the project team may want to consider if the project is too big and should be broken up into separate projects.

3.1.2 Writing a Business Purpose

The business purpose describes *why* CMS would fund the project. The business purpose is always related to the mission of the organization and/or financial concerns. CMS projects are often the result of legislative initiatives, which are therefore related to the mission of the organization. Upgrades and enhancements to systems often involve financial concerns, allowing the organization to take advantage of newer (more efficient) technology.

For example, the project team for UARS came up with:

> The business purpose is to comply with the provision set by Congress in Section 1011 of The Medicare Prescription Drug, Improvement and Modernization Act of 2003 (MMA).

It is possible that there are multiple business purposes for funding the project, and each should be written in complete sentences. When stating more than one purpose, ensure they don't compete. For example, complying with a law and updating technology to reduce maintenance costs are valid business purposes that do not conflict. If there are more than two business purposes, the project team may want to consider breaking the project up into smaller components.

3.1.3 Determining the Measures of Success

The measures of success list how the team shall be measured against the functional and business purposes. The measures of success must always be concrete and measurable. This can be especially challenging if the processes used in the existing system have never been measured. The measures must also tie back to the business and functional purposes. Some types of measures include:

- How much time particular business transactions take to complete
- Achievement of the strategic goal(s) identified in the BCA.

Determining what constraints are on the project will also help in defining the measures of success. Constraints mandate the way the system must be produced and are generally documented with the business requirements. Typical types of constraints include:

- Compliance: what legislative initiatives must be met?
- Solution: distinct from technological restraints, it defines the boundaries within which the problem may be solved and are absolutely non-negotiable.
- Schedule: how much time is allotted to build the system? Is there a window of opportunity that must be taken advantage of or deadlines that have to be met?
- Budget: how many dollars or people are available to the project?
- Current Implementation: if applicable, a description of the components (automated, mechanical, organizational and otherwise) of the current system.

For example, the project team for UARS came up with:

> UARS's measures of success are:
>
> - Compliant to Section 1011 of MMA
> - Documented Enrollment and Payment Process established by September 1, 2004
> - Establish audit trail process for medical and non-medical documentation
> - Be able to accept enrollment and claims by January 1, 2005; for the previous quarter (10/1/04 – 12/31/04)
> - After January 1, 2005, be able to accept enrollment and claims for all open periods.

3.1.4 Identifying the Stakeholders

Stakeholders of the project fall into four categories:

- Those who will pay for the system
- Those who will use the system

- Those who will be positively affected by the system’s implementation
- Those who will be adversely affected by the system’s implementation

It is important to list and discuss all categories to accurately determine each group’s concerns and needs. Identify those who will use the system by describing how they’ll use it and from where they’ll use it. It may also help to determine approximately how many users are in each grouping.

The stakeholders from UARS project were:

Payers

- CMS Management

Users (people and systems that interact with UARS)

- UARS Administrative Contractor
- UARS Payment Overseer Contractor
- CMS Management
- Oscar/UPIN (Medicare Provider Database)
- Grouper, MCE, OCE (Code Processing)
- Physician Fee Schedule, Ambulance Fee Schedule, Pricer (Medicare Payment Calculators)

  (GAO and IG will NOT be direct users of UARS. They will interact with CMS management to get the information they need.)

Positively Affected

- AMA
- American Hospital Assn
- Advocacy Groups

Adversely Affected

- Hospitals may have concerns about the administrative costs of INS investigations
- Hospitals that don’t participate; those that feel that the administrative costs are higher than what is reimbursed
- State Medicaid Agencies (if patient is eligible, need to seek State Medicaid first, before UARS payments)

It is important to consider how the entire user and support staff will be affected by the new system. Many times, while a new system will ultimately make a user more productive, it often places additional burdens or demands on support groups. Understanding how they are adversely affected can uncover new assumptions that may need to be addressed.

3.1.5 Determining the Project Risks & Assumptions

It is valuable to list the assumptions that the project team and even developers are making about things on which they will base their project planning decisions. Each assumption should be assessed and state the likelihood that it is correct. Identify a relevant list of alternatives, if an assumption does not happen.

Similarly, identify the project risks that the organization faces should the project, or some portion of it, fail. Assess the level of risk (high, medium, low), the consequences should the risk become a reality, and some methods for reducing the risk.

The UARS project team identified the following risks:

- Schedule: gathering requirements in June while policy wasn't even set, yet releasing RFP in August.
- Design/User Friendliness; not using UARS or the available funding dollars because the implementation of the system and the process is too complex
- Limited knowledge of system interfaces to the code and payment calculation systems listed above; these are implemented at the FIs and carriers, and therefore are implemented differently.

3.1.6 Assessing the Mission Profile

Successful completion of a project depends on several factors. Among these are the resources available to do the work, the proper scope of the work, the work being completed on schedule to meet business needs and a quality solution that is free from defects. However, there is always an inherent conflict between scope, resources, schedule and allowable defects.

The Mission Profile helps the team determine what is most important among the competing priorities, should a choice need to be made. The team is only allowed to select one of these as being of highest importance to the project. Ideally, only one of the four should be designated as second highest. Use a chart similar to the following to discuss the four priorities with the team:

Product Quality Dimension	Priority Level		
	Excel (High)	Improve (Medium)	Accept (Low)
Scope (features)			**X**
Schedule	**X**		
Defects		**X**	
Resources (manpower, budget)			**X**

After making the selections on the chart, document why the team made its selections. For example:

> Schedule is most important. It will drive whether the team uses some existing, proven systems vs. investing in new technology. Minimal scope must be reached (enrolling, submitting payment request, calculating payment, payment), but complexity and depth of each task is negotiable, based on time/budget.

3.1.7 Writing Business Requirements

Business Requirements are the high level features of the system to be developed. They specify at the most general level of detail 'what' the product must do to support the Business Purpose. These business goals address legislative mandates or strategic objectives and include the core functionality of the system (such as improved customer service, business efficiencies, business process reengineering, etc.). Each business requirement (e.g., “CWF shall validate all claims before payment is made.”) is supported by lower level functional and nonfunctional requirements and must be derived from the Business Purpose.

Use the following guidelines for writing Business Requirements:

1. Start every requirement with "The system shall…" to support the actions for a group of functional/nonfunctional requirements.

2. The requirement should be written in clear, concise English language only and not reference any data elements, value codes, or other such constructs found in functional requirements or system design documents.

3. The Business Requirement will represent the synthesis of the intent of multiple functional/nonfunctional requirements

Some of the Business Requirements from the UARS project include:

3.1.1 UARS shall implement its payment calculation using established Medicare processes to ensure the integrity of the payment calculation methodology.
This is why UARS must interact with Grouper, OCE, OCE for Indian Health, MCE, Pricer, UPIN, Oscar, Physicians Fee Schedule, and Ambulance Fee Schedule, and EDB.

3.1.2 UARS shall only interact with eligible providers.
An eligible provider is a physician, physician group, hospital, ambulance service, Indian Health Organization, or tribal organization that
- o Has a Medicare Provider Number, and
- o Performed the service in the fifty states or the District of Columbia.

3.1.3 UARS shall archive historical data indefinitely.

3.1.4 UARS’s design shall accommodate moving from legacy systems (e.g., UPIN and Oscar numbers) to updated standards (e.g., National Provider Identifier) without major modifications to UARS’s code.
The design shall be approved by CMS.

3.1.5 UARS’s design shall accommodate functioning with multiple sets of state allocations to accommodate multi-year reporting and multi-year processing.
That is, at the end of one fiscal year, while beginning another.

Note that Business Requirements are often generated from Legislation, Business Process Model (BPM) or Reengineering Efforts. Business Requirements can also specify standards or design criteria to which the project must adhere.

Refer to the Style Guide in Chapter 4 to understand how requirements should be phrased.

3.2 Requirements Management

3.2.1 Creating a Work Context Diagram

At the beginning of a system development project, it is a good idea to define the boundaries of the system to be studied/built. Within the requirements writing process, it is essential to know the scope of this system. The best tool for this purpose is the Work Context Diagram.

The context diagram defines the boundaries of the system being studied by showing how it connects to the outside world. The diagram is largely built before you begin to break the system into its functional pieces. It shows how the system under study connects to the systems surrounding it. These connections help convey the precise scope of study. The diagram demonstrates the scope of the systems analysis to the users. Without an agreed context, there is no way of telling if the correct system is being analyzed. There is no real place to start the analysis and no real place to stop.

The sample Work Context Diagram below shows all entities that will have knowledge of the system and that will interact with it. The direction of the arrows indicates which entity will *initiate* the event. After an event is initiated, there is usually a two-way communication. The Work Context Diagram's arrows simply show who begins the events. Note that it is possible for the system to initiate an event that interacts with itself. Timed processes such as an automated backup would fall under this category.

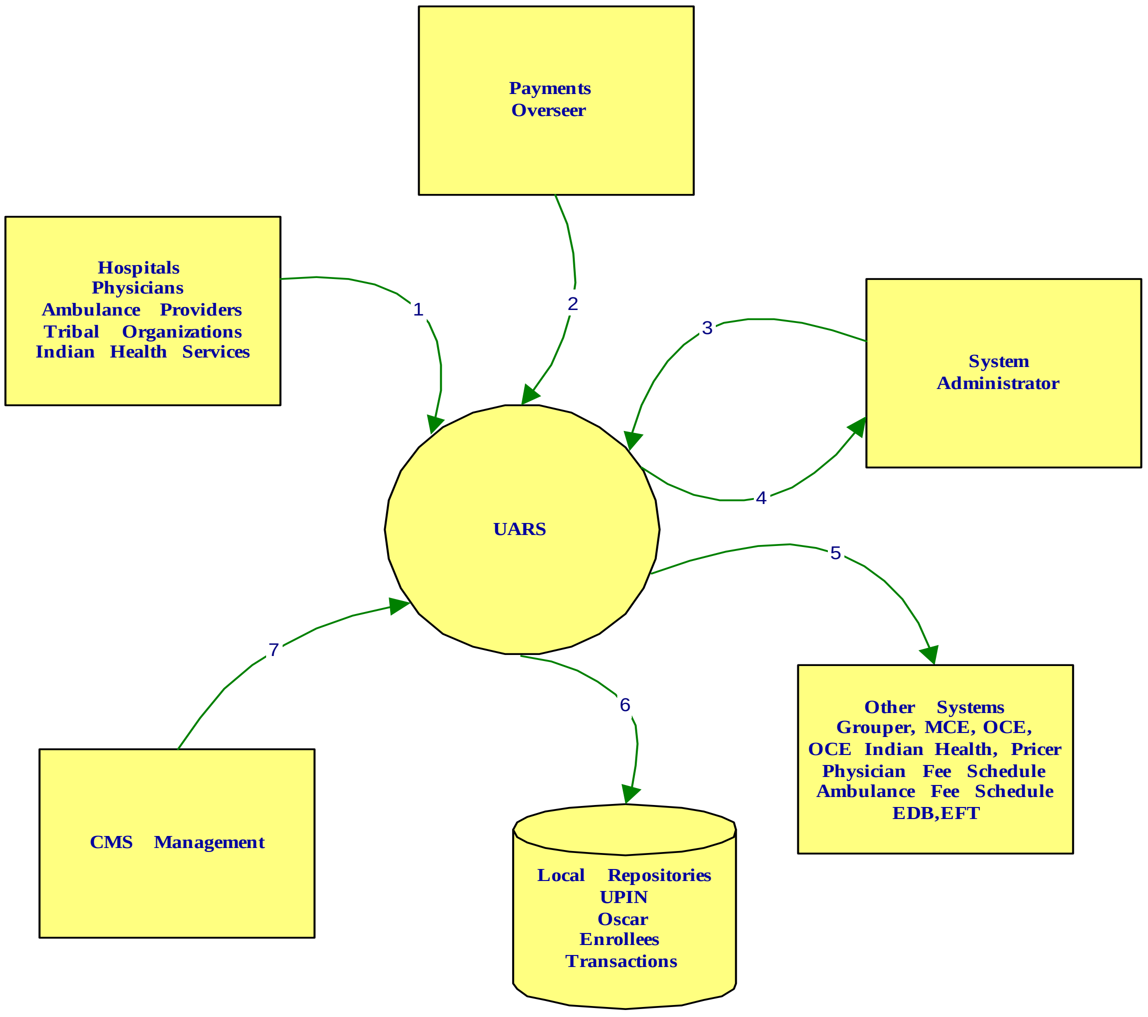


Figure 2 - Work Context Diagram

Using the Work Context Diagram the project team identifies the list of events initiated by that domain for each line. For UARS, these included:

Provider to UARS Events (line 1)

- Enroll as a Provider
- Submit a Claim
- Inquire the Status of a Claim
- Modify Information Entered at Enrollment
- Request Sending Forgotten Password

Be sure to write a description for each event so there is no ambiguity as to what the purpose of the event is by indicating who is doing what to whom. All events should begin with an action verb. For example, use ‘View Reports’ or ‘Create Reports’ instead of ‘Reports’ for clarifying what is expected. In the examples above, the Provider is doing something (submitting a claim) to the UARS system. The Project Owner is responsible for identifying these events with the assistance of Subject Matter Experts (be wary of losing focus by adding too much detail here).

From the events, scenarios will be generated that describe in more detail how the event will play out. Functional and nonfunctional requirements can later be generated for both events and scenarios.

3.2.2 Assessing Traceability

Traceability helps assure the completeness of implementation and assists in the validation of a system. It provides comfort that progress in the system is being made and it allows the Project Owner and/or stakeholder to assess the impact of proposed changes. It also ensures that all subsequent requirements stay within scope of the business requirements.

Traceability can help organize and maintain information when:

- Enhancing an existing system
- Creating a new system
- Managing contractors
- Running a project or other internal development effort

Using Traceability will improve a project's:

- Confidence that the objectives are being met
- Quality by meeting ***all*** the essential requirements
- Communication with development groups
- Accountability of subordinate organizations
- Change control through impact assessment
- Ability to verify that requirements are satisfied
- Ability to track the project in the early stages

3.2.2.1 Achieving End-to-End Traceability

Traceability information provides knowledge that allows the project owner and/or stakeholder to find dependencies either between requirements, or between the requirements and other documentation (e.g., system design).

Traceability starts with understanding the levels of requirements. At the top of the level is the business purpose of the system. This is broken down into business requirements which are further broken down into functional and nonfunctional requirements.

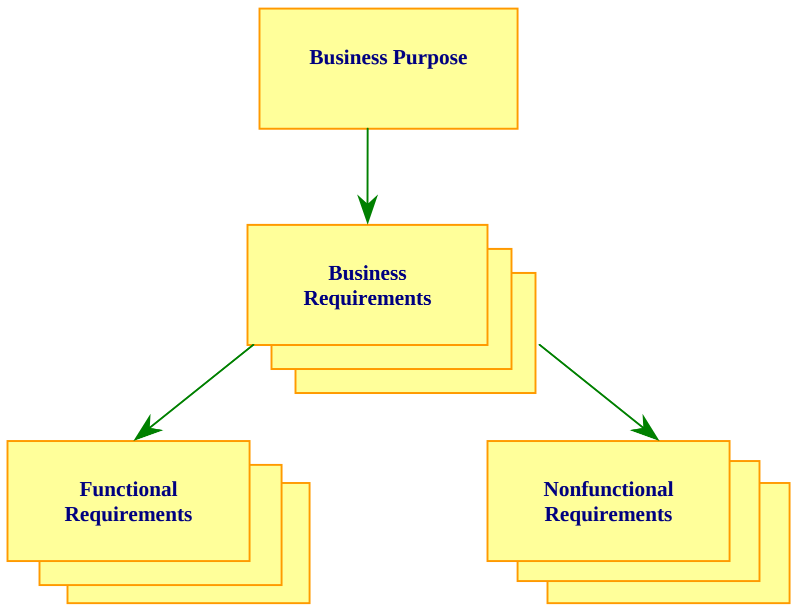


Figure 3 - Requirements Traceability

In this way, it is possible to trace functional and nonfunctional requirements back to business requirements, and it demonstrates that functional requirements exist for each and every business requirement. It is also quite common for a nonfunctional requirement to be dependent on a functional requirement. In those cases, it should be noted which requirement it is dependent on.

Listed below are the seven steps to achieving end-to-end traceability:

- **Uniquely identify each requirement**. Adopt a requirement numbering scheme where numbers are assigned sequentially and are not used twice within the same requirements set. Within the Dynamic Object Oriented Requirements System (DOORS), use a prefix of BP (for business purpose), BR (for business requirement), FR (for functional requirement) and NR (for nonfunctional requirement).
- **Assign each functional/nonfunctional requirement to one or more business requirement(s)**. Create a mapping to the next level down – those at the lower level will confirm this linking.
- **Project owner and/or stakeholder must review mapping between the functional/nonfunctional requirements and business requirements**. If requirements correspond, then create a link from the requirement back to the business requirement.
- **Perform traceability analysis, identifying all functional/nonfunctional requirements without a link**. This is called “checking for orphans” and “checking for childless” requirements.
- **Include maintenance of the end-to-end traceability matrix as part of your change control board activities**. After the traceability is established, the Project

Owner and/or stakeholder should keep the traceability current, as you would project costs, schedule and requirements documentation.

3.2.2.2 Using the Dynamic Object Oriented Requirements System (DOORS) for Establishing Traceability

Creating and maintaining a system is easier when there is a structured process in place. All requirements developed are housed in CMS' standard requirements management tool, as specified in CMS' Technical Reference Model. Currently, this tool is the Dynamic Object Oriented Requirements System (DOORS). If requirements are initially documented in another requirements management software tool, they must be able to be imported into DOORS.

At CMS, project teams and/or contractors use DOORS to maintain requirements and to establish traceability. DOORS allows users to create, track, trace and manage requirements for a system. Traceability links are included as attributes (or fields) in the DOORS database. When links are established in DOORS, the project owner and/or stakeholder will be able to generate traceability reports automatically. *Due to the nature of this tool, only* ***backward*** *traceability should be established.* However, while DOORS is a powerful repository for requirements, the Requirements Document is still critical for seeing the entire scope of the project.

In DOORS, a project owner and/or stakeholder will be able to establish traceability between test plans and functional/nonfunctional requirements, development specifications and functional/nonfunctional requirements and functional/nonfunctional requirements and business requirements. In order to create traceability, a project owner and/or stakeholder shall define the relationship between business requirements, functional/nonfunctional requirements, design entities and test plans. ***Backward*** traceability will ensure that the functional requirements ***satisfy*** all business requirements, development specifications ***satisfy*** all functional requirements and test plans and procedures ***validate*** all functional requirements. Traceability will also ***verify*** that the design was accomplished correctly. Shown below in Figure 4 is a pictorial representation with examples of requirement levels and requirement descriptions.

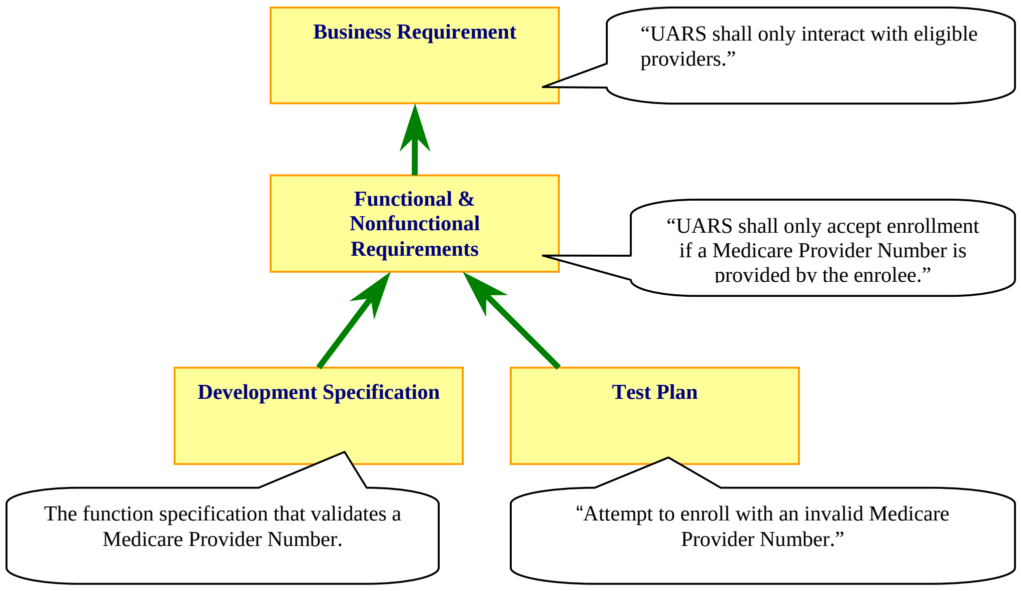

Business Requirement
"UARS shall only interact with eligible providers."
Functional & Nonfunctional Requirements
"UARS shall only accept enrollment if a Medicare Provider Number is provided by the enrolee."
Development Specification
Test Plan
The function specification that validates a Medicare Provider Number.
"Attempt to enroll with an invalid Medicare Provider Number."

Figure 4 - Requirement Levels

3.2.3 Baselining Requirements

Baselining requirements are critical in the system development life cycle. Baselines are established for the control of requirements during the analysis, design, testing and implementation phases of the life cycle process. Any document or specification that has been baselined can only be changed through formal change control procedures.

A baseline is a snapshot of the agreed-on requirements at a specific point in time. Each baseline is a uniquely identified, read-only copy of requirements. Uniquely identify each baseline to avoid confusion between drafts and other baselines, and between obsolete and current versions. Usually baselining occurs during the process of transitioning a requirements document under development into one that has been reviewed and approved.

The requirements are baselined prior to the informal review, prior to the formal walkthrough and after the acceptance (see next section).

3.2.4 Reviewing Requirements

The purpose of the review is to verify that the Requirements Document is acceptable. During this review process, discrepancies, issues and problems regarding the requirements document are submitted to the project owner via Review Item Discrepancy (RID) forms (see Appendix A - Templates). The project owner then categorizes the RIDs as open, rejected, or deferred. Open RIDs are sent to review participants. A review meeting is then held to discuss each open RID and to decide if changes to the Requirements Document are warranted. The culmination of this review is the approval of the Requirements Document.

The Project Owner conducts the review and usually asks team members, stakeholders, system engineers, testers, developers and sponsors to participate. The designers and developers check that the requirements are suitable for design and development. The walkthrough allows them to become more familiar with the requirements. Designers and developers can assess potential technical problems with the requirements. Testers are more likely to discover requirements that cannot be validated.

3.2.4.1 Major Roles within the Requirements Review

CMS uses two document review stages. The formal walkthrough is designed to provide more structure than the informal review.

3.2.4.1.1 Project Owner

During Informal Review Process: The team that was involved in drafting the requirements conducts this type of review. If an outside contractor developed the requirements, the contractor at the contractor's site performs the informal review process. The Project Owner is not involved in the contractor's informal review, except for receiving the updated requirements at the end of this review. *However, the project owner plans, coordinates the activities, and facilitates the review within the team for requirements developed in-house.*

During Formal Review Process: If necessary, the Project Owner is responsible for dividing the requirements into sections that can be discussed in a walkthrough period of 30-60 minutes. For additional project owner responsibilities within the formal requirements walkthrough, see the Moderator Role, below.

3.2.4.1.2 Moderator (Usually the Project Owner)

The Moderator plans, coordinates the activities and facilitates the formal walkthroughs. Prior to the formal walkthroughs, the moderator distributes a walkthrough agenda and a requirements section at least five days before each walkthrough meeting. During the walkthroughs, the moderator starts the meetings on time, encourages contributions from all participants and keeps each meeting focused on finding defects rather than resolving raised issues. After completing the walkthrough for a particular section, the moderator follows up on proposed changes with the requirements author(s). The moderator works to make sure the defects and issues discovered in the walkthrough are resolved.

3.2.4.1.3 Requirements Writer(s)

The Writer(s) is/are the individual(s) who created or maintained the requirements document. Here at CMS, the writer(s) actively participates in walkthroughs and often leads the discussions. At the walkthrough, the writer may explain their reasoning for writing the requirements the way they did, explain assumptions made and address any questions raised.

3.2.4.1.4 Reader

Assign a review participant the role of Reader during each section walkthrough. During the walkthrough, the reader can either quote directly or paraphrase the requirements section. Paraphrasing is important because by stating the requirement in his or her own words, the reader provides an interpretation that might differ from that held by other reviewers. Paraphrasing is therefore one way to reveal ambiguity or a possible defect.

3.2.4.1.5 Scribe

A Scribe records all issues raised and defects found during the walkthroughs. The scribe reviews aloud what was written to confirm the accuracy of the recorded information. Other review participants help the scribe capture the essence of each issue in a cogent way that clearly communicates to the requirement author the location and nature of the issue.

3.2.5 Tracking Change Control

A good requirements management process is necessary to effectively manage requirements throughout the development life cycle. Often after creating, reviewing and approving a good set of requirements, they will change. These changes will often occur, with little or no analysis of the effect of those changes or implications to cost or development schedules. This is an inevitable fact in the world of systems development. However, the ability to control such changes is crucial to any project.

There are many different ways to handle change, but the best way to manage change is to recognize change when it occurs and respond. Another way to handle change is to document it, assign the change a cost and prioritize it. This approach gives a simplistic and clear view of the project status when the need for a change occurs.

Another way to manage change during development is to use a tool to manage change. The change proposal feature in DOORS provides a systematic method for gathering proposed changes, reviewing and making decisions about those changes.

3.2.6 Enforcing Standards

Having writing standards is important for consistency throughout the organization’s documents. They only have value, however, if document writers adhere to them. It is ultimately the responsibility of the Project Owner to enforce good writing style in requirements documentation.

It is not uncommon, and not wholly unacceptable, that when a project must be delivered in a short time frame and the requirements need to get done, that certain standards are overlooked or deprioritized. The Project Owner is the one to determine what standards are most critical to the success of the project. It may be helpful to identify what will be acceptable (and unacceptable) to the requirements team before beginning the documentation.

If the Project Owner is also the Requirements Engineer, it is important to have someone else review the document. The Office of Information Services, the CIO’s Planning, Management, and Support Group, is a good place to find proofreaders who are well acquainted with writing standards and style guidelines.

3.3 Writing Requirements

Writing requirements is a difficult task, but the checklist provided below describes the purpose of requirements and can be summarized as follows:

- To communicate the needs of an user or organization
- To provide a solid foundation for the system or product
- To provide the first view of what the intended product must do
- To provide clear descriptions of how the system should perform
- To serve as a foundation for testing and user acceptance
- To provide a basis for design
- To provide clear goals and measurable objectives
- To show traceability back to the requirement sources

Typically the Requirements Engineer and the SME will work together closely to write requirements.

3.3.1 Creating Scenarios

Scenarios, also called Use Cases, describe the steps a user takes to satisfy an event required of the system. They are a convenient structure around which to organize requirements. There are often several scenarios associated with an event.

All the events listed in the work context diagram should have at least two or more scenarios associated with it. The first scenario is the primary scenario and is usually the most common way the Subject Matter Expert believes the scenario will play out.

Often it helps to diagram a scenario before writing the formal description. This type of diagram is considered to be a Business Process Model. Any diagramming methodology the team is familiar with is acceptable. Below is a sample diagram for a provider submitting a claim:

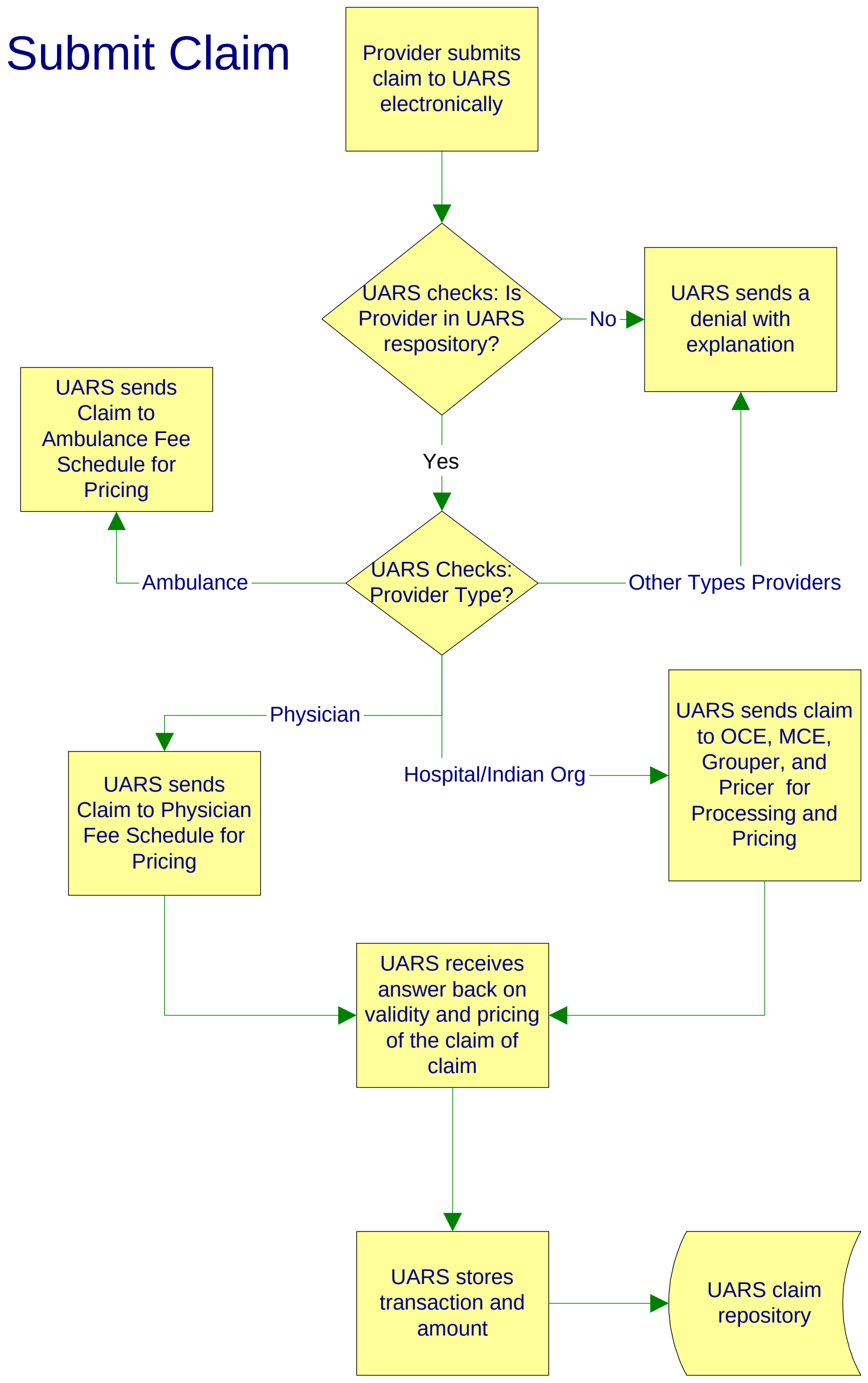

Submit Claim
Provider submits claim to UARS electronically
UARS checks: Is Provider in UARS respository?
No
UARS sends a denial with explanation
Yes
UARS sends Claim to Ambulance Fee Schedule for Pricing
Ambulance
UARS Checks: Provider Type?
Other Types Providers
Physician
Hospital/Indian Org
UARS sends Claim to Physician Fee Schedule for Pricing
UARS sends claim to OCE, MCE, Grouper, and Pricer for Processing and Pricing
UARS receives answer back on validity and pricing of the claim of claim
UARS stores transaction and amount
UARS claim repository

Figure 5 - Submit Claim Diagram

When documenting a scenario the following steps must be followed:

1) What are the preconditions that must be met before the event can occur? For example, if you are a system user, the preconditions to logging in are you must have a valid username and password set up by the system administrator.

2) What are the triggers that cause the event to occur? Multiple triggers will often indicate a need for multiple scenarios.

3) What is the expected result of the event? Knowing this will help you identify alternate scenarios as well as clarify the purpose of the scenario.

This flow diagram is documented in the Scenario Outline below. Note how to handle the diagrammed conditional situations along the way. In some cases, the condition results in the scenario terminating before the event is complete – this is normal. Note too that functional requirements are documented as part of the scenario in order to best understand the context of the requirement.

1 Provider to UARS Events and Requirements

1.1 Event: Submit A Claim

1.1.1 Scenario: Provider has a Claim to Submit

1.1.1.1 Trigger

Provider has a Claim to Submit

1.1.1.2 Precondition

Provider has Enrolled with UARS

1.1.1.3 Expected Result

The Status of a Claim is Accepted (pending payment) Denied or Paid

1.1.1.4 Steps

1.1.1.4.1 Provider submits the claim electronically to UARS

1.1.1.4.2 UARS checks to see if the provider is in UARS's Medicare provider repository

1.1.1.4.3 If the provider is not in UARS's provider repository (enrolled), UARS denies the claim by:

1.1.1.4.3.1 UARS logs the denial for reporting purposes

1.1.1.4.3.2 UARS notifies the provider of the denial

1.1.1.4.3.3 UARS's responsibilities are complete for this scenario

1.1.1.4.4 If the provider is in UARS's repository (enrolled), UARS continues with the following steps.

1.1.1.4.4.1 If it is a physician claim, UARS calculates the payment amount using the Physician Fee Schedule.

1.1.1.4.4.2 If it is an ambulance claim, UARS calculates the payment amount using the Ambulance Fee Schedule

.

.

.

You should expect to document complex events using several scenarios. Name the scenario, number it and include a description to help distinguish what is unique about each scenario.

3.3.1.1 Alternate Scenarios

Alternate scenarios are intended to document how unexpected exceptions are handled by the system. These are often discovered by going through each step in the scenario outline(s) and speculating on what could go wrong at that point. Exceptions could include system resource problems (e.g., connection is dropped or disk runs out of space).

For example:

2 Provider to UARS Events and Requirements

2.1 Event: Submit A Claim

2.1.1 Alternate Scenario: Communications Failure with Another System

2.1.1.1 Trigger

Provider has a Claim to Submit

2.1.1.2 Precondition

Provider has Enrolled with UARS

2.1.1.3 Expected Result

The Transaction is Suspended.

2.1.1.4 Steps

2.1.1.4.1 Provider submits the claim electronically to UARS
2.1.1.4.2 No connection to mainframe is established
2.1.1.4.3 UARS saves the transaction as suspended
2.1.1.4.4 UARS notifies the administrator that a suspended transaction exists

3.3.2 Writing Requirements

As the scenarios are fleshed out, requirements should be identified along the way. Requirements will always clearly state who is doing what to whom. They identify *what* is expected of the system. For example:

UARS shall check the Medicare Provider Number against its local repository of provider numbers.

Requirements should always be stated in the form of what is required of the system, not what is required of the user. For example, after writing a log in scenario the following requirements may apply:

1. The system shall require a user to change their password monthly.
2. The system shall require that a user not use the same password twice in a row.

This makes it clear what is expected from the system as well as the conditions/constraints that are imposed on the capability. Contrast requirements 1 and 2 above with the following:

a) The user will change his password monthly.
b) A user cannot use the same password twice in a row.

Requirements a) and b) sound like policies not functional requirements.

When documenting a requirement, it is also helpful to note the following:

- What is the purpose of the requirement?
- Does the requirement conflict with another requirement? If it does, there may be some additional scenarios or a business decision that needs to be made.
- Does the requirement depend on another requirement? If it does, does the scenario account for this?
- Is the requirement modular enough so that it can be changed without excessive impact on the system?
- What is the precedence and criticality of the requirement? Low precedence/critical requirements could be postponed for later releases should the schedule become a problem. Requirements may be rated as Essential, Conditional (enhances final system) or Optional (may not be worthwhile).
- Track the history of the requirement. When was it approved and by whom? When was it changed? When was it deleted? Who acted on it? What was the rationale?

Avoid creating requirements that specify a particular design or implementation method (unless there is a policy that constrains these options). For example, this requirement defines how the system will be designed, it is a 'how' statement:

> The System shall use a Web interface.

While this might be a valid requirement if the organization's policy is to use Web applications only, the more acceptable way to write this would be:

> The system shall allow access for remote, non-network users.

Finally, ensure that the requirement is feasible. Can it be done within the scope of budget and time allotted to the project? If not, several options exist:

- Throw out or postpone the requirement (for a later release)
- Change the requirement or its parent dependency
- Cancel the project (assuming the requirement is critical)

Refer to the style guidelines in Chapter 4 for more information on phrasing requirements.

3.3.2.1 Writing Nonfunctional Requirements

Nonfunctional requirements will detail the properties of the system. In the process of documenting these requirements, it's not unusual to uncover additional functional requirements. Nonfunctional requirements include:

- Required States and Modes: identify the various states or modes in which the system can exist
- Performance Requirements: this is the expected response time of critical system functions or intervals at which functions are expected to trigger.

- Security Requirements: identify the security measures (management, operational, and technical) that provide adequate protection against threats and vulnerabilities to the system based on its system security level
- Human-factors Engineering (Ergonomics) Requirements: these are also referred to as Look and Feel requirements. They may take the form of interface sketches or more detailed scenario models. They will help the designer understand how the user works with the functionality of the system. Identify areas that need concentrated human attention and are sensitive to human errors. Also consider manual operations to be performed that affect the system.
- Design Constraints and Qualification Requirements: identify what laws the system must uphold as well as any applicable standards the system must comply with. Consider any organizational policies that will affect the operation or performance of the system. List any existing components that could be reused.
- System Quality Characteristics: quantify the expected accuracy of the results produced by the system such as precision or units of measure.
- Internal Data Requirements: identify the essential objects/entities/classes that are germane to the system. This could be in the form of a first cut data model or a domain model. Consider integrity constraints, as well as retention and data replication requirements. Determine if any data conversion is required.

3.3.3 Determining Pass/Fail Statements

The Pass/Fail Statement should be a statement that describes how we will know if the requirement was met. Include the Pass/Fail Statement with the description of the requirement for easy reference. For example:

- The system shall require a user to change their password monthly. (requirement).

  Upon login, the system will check to see if more than 1 month has past since the password was last changed and prompt the user to choose a different password when that occurs. (pass/fail statement)

The Pass/Fail Statement may also state what will happen if the requirement is not met. For example:

- The system shall require that a user select a different password when required to change their password. (requirement)

  If the user enters their existing password as their new password, the system will notify them that their entry is invalid and request a different password. (pass/fail statement)

Writing the Pass/Fail Statement now not only helps the designer understand the requirement better, it makes writing test procedures much simpler. It may also help determine whether this is a requirement or an assumption. For example, consider the following requirement:

- The system shall utilize the user's network username as their logon id.

If your system lacks the ability to test whether or not the user's network username actually exists, what you have uncovered is an assumption rather than a requirement.

Writing the Pass/Fail Statement can also help you see if there are extra steps (or a whole scenario) that were overlooked and if your requirement is ambiguous.

4 Style Guidelines

The process of documenting requirements is a never ending cycle. When using thorough writing techniques, it is quite easy to spend a considerable amount of time eliminating ambiguity in order to ensure all assumptions, as well all the requirements, have been identified. This process inevitably leads to new requirements that should be examined as well. It falls on the Project Owner to decide what level of detail and investigation is required for the Requirements Document before a diminishing amount of return is gained from effort put forth.

This section of the guide discusses what might be considered a 'middle level' investigation of requirements. Proper layout, form and phrasing of requirements can help ensure the document is complete and clear. For an even deeper level of requirements specification, please refer to Appendices C and E.

4.1 Writing Style

CMS uses the Government Printing Office (GPO) Style Manual as the basis for writing. The GPO Style Manual is available through the following URL: http://www.gpoaccess.gov/stylemanual/browse.html. In some cases, the procedures depart from GPO rules. This guide takes precedence over conflicting GPO rules.

A properly structured requirement must be a complete sentence that has one and only one interpretation. Any individual who writes requirements should avoid using phrases, single words or a collection of acronyms to express requirements. Each requirement should consist of a subject, verb and predicate as noted in the requirement written below:

Requirement example:

> The Medicare Managed Care System shall assign a disenrollment date for a beneficiary in a Managed Care Organization when a termination date for the beneficiary's Part B eligibility is received.

The subject of the requirement is "The Medicare Managed Care System" and the predicate is "assign a disenrollment date for a beneficiary in a Managed Care Organization when a termination date for the Beneficiary's Part B eligibility is received." The subject of any requirement should imply ownership and shows the entity to which the requirement should be related. The predicate should be an action phrase or expression of something that must be accomplished for, by or to the subject. The verb used in requirements is always 'shall'. This implies that the requirement must be adhered to.

Keep the following in mind:

- Write in such a way that the requirement does not pose a specific solution. It should always be design independent.
- Write in such a way that there is one and only one interpretation of the requirement.
- Increase readability by: using grammatically correct language and symbology; using simple words/phrases/concepts; define unique words or symbols.

4.2 Eliminating Ambiguity

All natural languages (e.g., English) are inherently ambiguous. Use the following techniques to assist in making the requirements document clearer:

1. **Multiple requirements in the same requirements statement –** Do not use conjunctions (*and, or, with, also*). However, if the requirement has two or more conditions that must be met simultaneously, then conjunctions may be used.
2. **Let-out or escape clauses –** Do not use the following expressions: *if, when, but, except, unless, possibly, eventually or although.*
3. **Rambling –** Avoid long sentences.
4. **Mixing different kinds of requirements –** Do not mix business requirements with functional requirements in the same section of the document.
5. **Speculation –** Avoid speculative words such as *usually, generally, often, normally, or typically.*
6. **Vague terms –** Do not use unverifiable, subjective terms (typically adjectives). Look for:
   - adjectives related to intrinsic characteristics: *clear, well, easy, strong, weak, good, bad, efficient, low, user-friendly, flexible, effective etc.*
   - adjectives related to environmental characteristics: *useful, significant, adequate, fast, slow, versatile, a minimum, as applicable, as appropriate, etc.*
   - adjectives related to time characteristics: *old, new future, recent, past, today's, normal, timely, etc.*
   - adjectives related to location characteristics: *near, far, close, back, in front, etc.*
   - adverbs: *approximately, be able to, be capable, but not limited to, capability of/to*

   Replace these with actual measures.
7. **Suggestions or possibilities –** Do not use terms such as *can, optionally, may, might, should, ought, could, perhaps, or probably*.
8. **Wishful thinking –** Avoid phrases such as 100% reliable, totally safe, handles all unexpected failures, pleases all users, runs on all platforms, never fails, or fully compatible to all future situations.
9. **Negative statements –** Identification of duplicate requirements is facilitated if requirements are consistently written in the affirmative. In addition, negative statements can be impossible to test.
10. **Jargon –** Jargon often uses terms that all readers are not familiar with. This includes acronyms, abbreviations and abstract words or phrases.
11. **Passive Voice -** Verbs may be either *active* (The executive committee approved the new policy) or *passive* (The new policy was approved by the executive committee) in voice. In the active voice, the subject and verb relationship is

straightforward. In the **passive voice**, the subject of the sentence is acted upon by some other agent or by something unnamed. Active voice is stronger and engages the reader.

12. **Implicit Subjects –** an implicit subject is inherently vague and open to interpretation. Implicit subjects are found in sentences where the subject:
    - contains a demonstrative adjective indicated by *this, these, that, those*
    - is expressed by means of pronouns indicated by *it, they*
    - is specified by a preposition such as: *above, below*
    - is specified by an adjective such as: *previous, next, following, last, first*

    For example: *This counter* shall be incremented by 1 each time an update occurs.
13. **Under referencing** – documents or entities not explicitly defined when referred to can create ambiguity. Look for phrases such as: *according to, on the basis of, relative to, compliant with, conform to, etc.* For example: The software shall be designed according to the rules of object oriented design.
14. **Multiple Terms for the Same Subject –** Do not use different terms for the same subject. Agree on a uniform, or standard, term to be used to describe the same thing. For example in the Medicare Program, the "standard" term that describes a patient, a subscriber, or a beneficiary is "beneficiary." The Glossary located within the Medicare.gov Website may be helpful in standardizing terms.

Appendix A – Templates for the Requirements Process

These templates are provided to help ensure completeness and to help standardize how requirements are documented and tracked. However, the unique characteristics of each project may necessitate the tailoring of templates within this appendix.

A.1 Business Requirement Template

This template is used to document business requirements.

A.1.1 Directions

Requirement Number – Provide a unique identifying number. This number will be important when establishing traceability.

Requirement (the "shall" statement) – Provide a complete sentence using the word "shall" that describes the capability needed by the business to meet their objective.

Purpose (optional) **–** A brief rationale for the requirement

Source (optional) – The legislation or strategic objective document
If work has begun on the Requirements Document (RD), the source may be included in the Business Purpose. (See Section 1.4 of the RD, shown in Appendix B).

A.1.2 Example

Requirement Number – BR001

Requirement - UARS shall implement its payment calculation using established Medicare processes to ensure the integrity of the payment calculation methodology.

Purpose - This is why UARS must interact with Grouper, OCE, OCE for Indian Health, MCE, Pricer, UPIN, Oscar, Physician Fee Schedule, and Ambulance Fee Schedule, and EDB

Source - Medicare Modernization Act of 2003

A.1.3 Blank Business Requirement Template

Requirement Number –

Requirement (the “shall” statement) –

Purpose (optional) **-**

Source (optional) **-**

A.2 Functional/Nonfunctional Requirement Template

This template is used to document functional and nonfunctional requirements.

A.2.1 Directions

Requirement Number – Provide a unique identifying number. This number will be important when establishing traceability.

Requirement (the "shall" statement) – Provide a complete sentence using the word "shall" that describes the action or expectation of what the system will do (functional requirement), or the behavioral property the system must have.

Purpose (optional) **–** A brief rationale for the requirement

Use Case Number (optional) **–** If the requirement is derived from a specific use case, please provide the use case number.

Validation Measure (Demonstration, Test, Analysis, Inspection, Special Qualification Method, or Not Applicable) – Provide a description of how the requirement will be validated.

Pass/Fail Statement – A testable statement that describes how to know if a requirement was met.

Associated Business Requirement – Provide the business requirement number of the associated business requirement.

Dependent Requirement – List any requirements that this requirement is dependent on.

Priority (Essential, Conditional, Optional) – Provide the priority of implementing this requirement from the user perspective. Essential means "must have," Conditional means "like to have," and Optional means "could do without".

Comments – Please provide any comments if necessary.

Note[1]: Nonfunctional requirements will be a subset of the RD. Be sure to include which category (from Section 3.3.2.1) the nonfunctional requirement falls into.

Note[2]**:** This template conveys the recommended information to be captured in each requirement. Project owners are encouraged to add any other information that is appropriate for their projects. If the requirements are initially written in software other than DOORS, the formatting of this information is left to the discretion of the project owner. (A CMS-tailored template is already in DOORS.) Multiple requirements may be displayed on one page. A tabular format (e.g., as in MS Word or MS Excel tables) may also be used.

A.2.2 Example of a Functional Requirement

Requirement Number –FR001

Requirement (the "shall" statement) – The Customer Service Representative shall be able to confirm that a check has not been cashed.

Purpose – This allows the Customer Service Representative to know if the customer has already received payment.

Use Case Number (optional) – MCSC Next Generation Desktop Use Case 5: Initiate Check Reissue

Validation Measure (Demonstration, Test, Analysis, Inspection, Special Qualification Method, or Not Applicable) – Test

Pass/Fail Statement – Examining the current status of the check will indicate if the check has been cashed.

Associated Business Requirement – BR001

Dependent Requirement – None

Priority (Essential, Conditional, Optional) – Essential

Comments – Implement in Release 2

A.2.3 Blank Functional/Nonfunctional Requirement Template

Requirement Number –

Requirement (the “shall” statement) –

Purpose -

Use Case Number (optional) –

Validation Measure
(Demonstration, Test, Analysis, Inspection, Special Qualification Method, or Not Applicable) –

Pass/Fail Statement -

Associated Business Requirement –

Dependent Requirement -

Priority (Essential, Conditional, Optional) –

Comments –

A.3 Baseline Template

This template is used to document the official baseline of internally and externally developed requirement documents.

A.3.1 Directions

Each analyst will be responsible for completing this document prior to baselining requirements in DOORS. The baseline template will allow the user of this document to record the following:

- The type of document to be baselined and the baseline date
- An overview of the document to be baseline
- Any previous baselined version and the systems development effort
- An overview of any changes made to this release that differs from a previous version
- The total number of requirements to be baselined
- Confirmation signatures

A.3.2 Blank Template

DEPARTMENT OF HEALTH & HUMAN SERVICES
Centers for Medicare & Medicaid Services
7500 Security Boulevard
Baltimore, Maryland 21244-1850

CMS
CENTERS for MEDICARE & MEDICAID SERVICES

OIS/PMSG

This document contains information relating to baselined documents. Please fill out each item below before attempting to make changes in the DOORS Database.		
Document Type [ ]		
Baseline Date:		
Brief Description of the Document to be Baselined:		
Previous baseline version: (if applicable)	**Development Effort:**	
Total Number of Requirements to be baselined:		
Confirmation that all requirements have been reviewed by subject matter experts	[signature of analyst]	[date]
Confirmation that all requirements have been approved and signed off	[signature of Government Task Lead]	[date]
Approval to baseline (Project Officer)	[signature of project officer]	[date]

A.4 Review Item Discrepancy Template

A.4.1 Directions

This template should be used to support the CMS informal requirements review process. The RID template is used to document discrepancies that are found during the informal review of requirements documentation.

Date: Date the RIDs are produced
Originator: Author of the discrepancies
Project Name: CMS Project Name
Document Title: Title of the document where discrepancies are found.
Document Version: Version of the requirements document

This information appears once at the beginning of the list of RIDs.

RID Number: Generate these numbers in sequence.

Requirement Number(s)
or Location within RD: Unique requirement identifier(s) or specific location of problem.

Problem Description: Describe the problem in detail.

Recommended Solution: Describe how the problem can be corrected.

Project Owners Response (open, rejected, deferred): (to be completed by project owner)

RID Review Decision (accepted, accepted with modifications, rejected): (to be completed during the review meeting for user/business requirements or by the project owner during the informal system requirements review.)

A.4.2 Example

Date: 4/15/02
Originator: Jane Doe
Project Name: Regional Office Tracking and Reporting System (ROTRS)
Document Title: *Preliminary Working Draft* ROTRS User Requirements
Document Version: Hard copy handout from 4/11/02 meeting (no version number indicated)

RID Number: 1

Requirement Numbers: FR0190, FR0200, FR0310, FR0340, FR0360, FR0480, FR0490, FR0550, and FR0730,

Problem Description: Multiple requirements are contained in one requirement statement. Combining two or more thoughts (e.g., functions) in a requirement statement makes development and testing more prone to errors of omission.

Recommended Solution: Split the statement into the appropriate number of stand-alone requirement statements. For example, FR0190 states "The RO analyst shall be able to take the background *Reconsideration Information* over the telephone and record it for each Expedited Reconsideration Request, including hospital discharge cases; and annotate *Expedited Reconsideration* and *Expedited Reconsideration Hospital Discharge Case* where applicable." Clarity, verifiability, and traceability would be enhanced if this one requirement were written as follows:

1. The RO analyst shall be able to take the background *Reconsideration Information* over the telephone and record it for each Expedited Reconsideration Request, including hospital discharge cases.
2. The RO analyst shall be able to annotate *Expedited Reconsideration,* where applicable.
3. The RO analyst shall be able to annotate *Expedited Reconsideration Hospital Discharge Case,* where applicable.

Project Owners Response (open, rejected, deferred): Open

RID Review Decision (accepted, accepted with modifications, rejected): Accepted

---

RID Number: 2

ETC.

A.4.3 Blank Template

Date: __/__/__
Originator:
Project Name:
Document Title:
Document Version:

This information appears once at the beginning of the list of Review Item Discrepancies.

RID Number:

Requirement Number(s)
or Location within Requirements Document:

Problem Description:

Recommended Solution:

Project Owners Response (open, rejected, deferred): (to be completed by project owner)

RID Review Decision (accepted, accepted with modifications, rejected): (to be completed by project owner)

Appendix B - Requirements Document Template

To help ensure that the sample Requirements Document is clear, only a portion of the UARS Requirements Document is included here. This sample represents a complete description of a particular event starting from the Functional Purpose through the Functional/Nonfunctional Requirements.

Be aware that the unique characteristics of each project may necessitate the tailoring of the following Requirements Document Template.

B.1 Directions

Most of the sections in the Requirements Document are explained above in Chapter 3. Refer to the section listed below for more information.

Functional Purpose: Section 3.1.1
Business Purpose: Section 3.1.2
Measures of Success: Section 3.1.3
Stakeholders: Section 3.1.4
Project Risks & Assumptions: Section 3.1.5
Mission Profile: Section 3.1.6
Work Context Diagram: Section 3.2.1
Business Requirements: Section 3.1.7
Functional/Nonfunctional Requirements: Section 3.3.2
Event Scenarios: Section 3.3.1

B.1 Example

The example of a Requirements Document begins on the next page.

1 Introduction

1.1 Document Intent

This document shall outline the necessary functionality that the new undocumented alien reimbursement system will be responsible for implementing. This document lists the requirements of the system, requirements of the users, along with sample scenarios. The scenarios will help clarify the process required for the new system.

This document assumes that its audience is either a CMS employee or a Medicare Contractor who is familiar with claims processing for both Parts “A” and “B”. This document does not contain many of the systems requirements normally part of a requirements analysis, because the Medicare Contractors developed the interaction between systems. Therefore, it is assumed that the contractors have this information.

During initial discussions, it is often helpful to give a new, desired system a name. This helps the team during discussions differentiate between the new, desired systems and the limitations of any current systems. The team named the new system “UARS”, short for “Undocumented Alien Reimbursement System”. This document will refer to the new, desired system as “UARS” and on occasion, “Marvin”.

1.2 Definitions

1500	Medicare claim form for physicians
Ambulance Fee Schedule	Table used to calculate payment to ambulance services (See Ambulance Fee Schedule Sample File)
AMA	American Medical Association
CMS	Centers for Medicare & Medicaid Services
CPT Codes	Code to calculate physician payment; related to treatment/service provided by the physician
CWF	Common Working File; holds social security numbers and of beneficiaries, among other beneficiary information; EDB is one source that updates this data.
DRG	Diagnosis Related Group
EDB	Enrollment Database; stores Medicare Beneficiary Information
EIN	Employer’s Identification Number; Federal Tax ID Number
EMTALA	Section 1867 of the SSA which requires hospitals and physicians to provide emergency aid
EOY	End of Year; the end of the Government fiscal year - 09/30
FI	Fiscal Intermediary, Processor of Medicare Claims
FISS	Fiscal Intermediary Standard System (see FISS Logical Architecture)
GAO	General Accounting Office
Grouper	Inpatient Code Processing System; used to edit hospital claims

IG	Inspector General
INS	Immigration and Naturalization Service
MCE	Medicare Code Editor; used to edit hospital claims
MMA	Medicare Prescription Drug, Improvement and Modernization Act
NPI	National Provider Identifier
OCE	Outpatient Code Editor; used to edit all outpatient claims, which leads to correct bundling and pricing
Open Period	The period for which UARS will accept and process claims. UARS recognizes four quarters during the federal fiscal year: (1) 10/01-12/31 (2) 01/01-03/31 (3) 04/01-06/30 (4) 07/01-09/30 UARS shall accept and process claims for services provided during the current quarter, in addition to the previous quarter. In other words, providers have until the end of the subsequent quarter to submit a claim.
Oscar	Medicare Provider System and Numbering System for Hospitals, etc.
Physicians	Physicians or Physicians Groups
Physician Fee Schedule	Table used to calculate payment of physician fees based on CPT Codes (See Physician Fee Schedule Sample File)
Pricer	Medicare payment calculation system for hospitals
States	50 U.S. States plus the District of Columbia
UB-92	Medicare claim form for hospitals
UPIN	Unique Personal Identification Number; physician and Ambulance (plus others) identification number and other data. (See UPIN Text File)

1.3 Functional Purpose

The functional Purpose describes *what* UARS shall do.

The functional purpose is to build an enrollment and payment processing system to reimburse health care providers (hospitals, physicians, ambulance providers, Indian Health Services and tribal organizations) for emergency treatment of undocumented aliens.

1.4 Business Purpose

The business purpose describes *why* CMS would include funding in their budget for UARS. The business purpose is always related to the mission of the organization and/or financial concerns.

The business purpose is to comply with the provision set by Congress in section 1011 of the Medicare Prescription Drug, Improvement and Modernization Act of 2003 (MMA). No current system exists.

1.5 Measures of Success

The measures of success list how the UARS team shall be measured against the business and functional purposes. The measures of success must always be concrete and measurable.

UARS's measures of success are:

- Compliant to Section 1011 of MMA
- Documented Enrollment and Payment Process established by September 1, 2004
- Established audit trail process for medical and non-medical documentation
- Be able to accept enrollment and claims by January 1, 2005; for the previous quarter (10/01/04 – 12/31/04)
- After January 1, 2005, be able to accept enrollment and claims for all open periods

1.6 Stakeholders

Stakeholders for UARS fall into four categories:

- Those who will pay for the system
- Those who will use the system
- Those who will gain an advantage by the system's implementation
- Those who will lose an advantage by the system's implementation

It is important to list and discuss all categories to accurately determine each group's concerns and needs.

Payers

- CMS management

Users (people and systems that interact with UARS)

- UARS Administrative Contractor
- UARS Payment Overseer Contractor
- CMS Management
- Oscar/UPIN (Medicare Provider Database)
- Other Systems, such as Grouper, MCE, OCE (Code Processing)
- Physician Fee Schedule, Ambulance Fee Schedule, Pricer (Medicare Payment Calculators)

  (GAO and IG will NOT be direct users of UARS. They will interact with the CMS management to get the information they need.)

Positively Affected

- AMA
- American Hospital Assn.
- Advocacy Groups

Adversely Affected

- Hospitals that may have concerns about the administrative costs of INS investigations
- Hospitals that don't participate; those that feel that the administrative costs are higher than what was reimbursed
- State Medicaid Agencies (if patient is eligible, need to seek State Medicaid first, before Marvin payments)

1.7 Risks & Assumptions

- Schedule; time constraints may prohibit detailed requirements or some desired features form being implemented, as well as the ability to perform a thorough test suite
- Design/User Friendliness; not using UARS or the available funding $$$ because the implementation of the system and process is too complex
- Limited knowledge of system interfaces to the code and payment calculation systems listed above; these are implemented at the FIs and carriers, and therefore are implemented differently

1.8 Mission Profile

There is always an inherent conflict between scope, budget available, schedule and allowable defects. The mission profile helps the team determine what is most important, should a choice need to be made. The team is only allowed to make one check in the "High" importance column, so that everyone agrees on what is the highest priority.

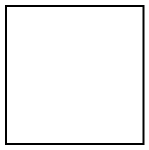
Product Quality Dimension	Priority Level		
	Excel (High)	Improve (Medium)	Accept (Low)
Scope (features)			X
Schedule	X		
Defects		X	
Resources (manpower, budget)		X	

Schedule is most important. It will drive whether the UARS team uses some existing, proven systems vs. investing in new technology, etc. Minimal scope must be reached (enrolling, submitting payment request, calculating payment, payment), but as complexity and depth of each task is negotiable, based on time/budget.

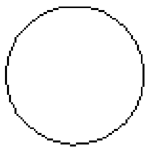
2 Overview

2.1 Context

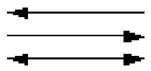
Below is a key to the types of symbols used in a context diagram.



Domains of Interest are subject matter areas, or activities, that have an effect on the business you are analyzing. May have slightly rounded corners or shaded to indicate different relationships



The Work Context, this is the business that is to be analyzed



Joins a Domain of Interest to the Work Context and why it is relevant to your study. This also shows the primary flow of data.

2.2 Work Context Diagram

Figure 2-1 on the following page shows the work context diagram for UARS. The work context diagram shows all entities that will have knowledge of UARS and that will interact with it. The direction of the arrows indicates which entity will *initiate* the event. For instance, since there is only one arrow going from CMS Management to UARS, you can assume that UARS shall NOT automatically generate an event, like reports, to CMS management. CMS Management must initiate the event by requesting reports. After an event is initiated, there is usually two-way communication. The Work Context Diagram's arrows simply show who begins the events.

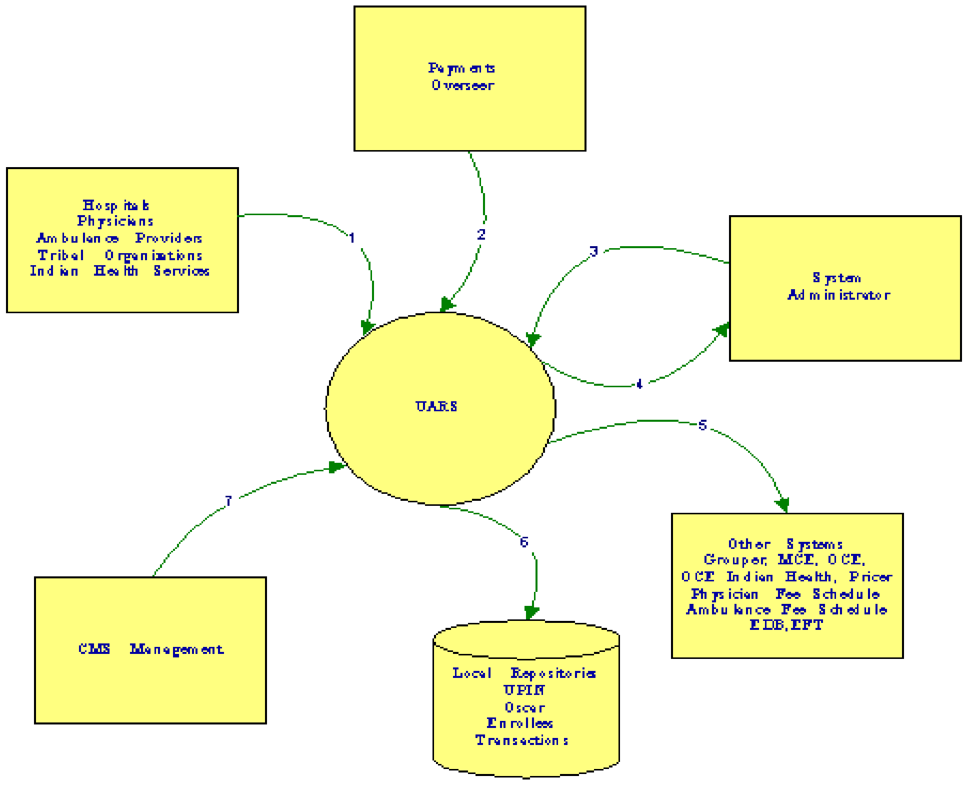


Figure 2-1

While Section 3 lists the business requirements for UARS, Section 4 and following sections lists the functional requirements for the events that transpire between UARS and the entities seen in Figure 2-1. Each event lists the requirements specific to that event, scenarios and alternate scenarios. Scenarios show how many ways the event can occur. For instance, a log in event may have two scenarios: local and remote. An alternate scenario is a scenario that gives an unexpected result or is a scenario “gone wrong”. For instance, an error situation the UARS must handle.

Requirements are also listed in other sections. Where they are listed infers how they affect UARS. For instance, Standards and Design requirements affect all of UARS, while requirements listed in an event section pertain only to the event.

3 Business Level Requirements

3.1 Business Requirements

3.1.1 UARS shall only interact with eligible providers (BR-0008).
An eligible provider is a physician, physician group, hospital, ambulance service, Indian Health Organization or tribal organization that

- Has a Medicare Provider Number, and
- Performed the service in the fifty states or the District of Columbia

4 Provider to UARS Events and Requirements:

4.1 Enroll As a Provider

4.1.1 Requirements

4.1.1.1 UARS shall only accept an enrollment if a Medicare Provider Number is provided by the enrollee (FR-0022).

UARS's list of providers could be out-dated since it is updated only periodically, which will mean that some valid providers may not be in UARS's provider list databases at the time they wish to enroll. Therefore, UARS shall only check to see if it is valid against its local repository, and if not, shall check to see if the number is well formed.
Precedence: Critical.
Pass/Fail Statement: The system rejects an invalid or malformed provider number.

4.1.1.2 UARS shall only accept enrollment of eligible providers (NR-0024).

Eligible providers are providers that

- Provide UARS with a Medicare Provider Number
- Are a physician, physician group, ambulance service, hospital, Indian Health Service, or Indian tribal organization

Precedence: Critical.
Pass/Fail Statement: An eligible provider cannot enroll.

4.1.1.3 UARS shall only save enrollment data if it is complete (NR-0025)

If an enrollee begins enrollment, then realizes they don't have all the information they need, such as UPIN numbers or bank routing information, UARS shall not save the information when they exit.
Precedence: Conditional
Pass/Fail Statement: An incomplete enrollment transaction is not saved

4.1.1.4 UARS's enrollment process shall send the provider written confirmation of the enrollment (FR-0029)

UARS's process shall send the confirmation to the UPIN address (if validated; the entered address otherwise), and include the UPIN number, and EFT information.
Precedence: Critical
Pass/Fail Statement: the provider receives a confirmation after enrollment.

4.1.2 Scenario: Medicare Provider Wishes to Enroll with UARS

4.1.2.1 Trigger

Medicare Provider wishes to Enroll with UARS

4.1.2.2 Precondition

Provider has a Medicare Provider Number

4.1.2.3 Expected Result

Provider is Enrolled with UARS

4.1.2.4 Steps (See Enrollment Process Flow Diagram)

4.1.2.4.1 Provider goes to UARS's secure web site
4.1.2.4.2 Provider enters the following information
4.1.2.4.2.1 Select type of provider:
4.1.2.4.2.1.1 Hospital
4.1.2.4.2.1.2 Physician/Physician Group
4.1.2.4.2.1.3 Ambulance Service
4.1.2.4.2.1.4 Indian Health Service
4.1.2.4.2.1.5 Tribal Organization
4.1.2.4.2.2 Medicare Provider Number (entered twice to make sure entered correctly)
4.1.2.4.3 When UARS gets the Provider Number, UARS shall check the number against its local repository of provider numbers
4.1.2.4.4 If UARS finds a match, UARS shall display the information it has about the provider.
4.1.2.4.4.1 UARS shall not allow modification of this data (since it must sync up to UPIN and Oscar)
4.1.2.4.4.2 UARS shall allow the enrollee to enter another number in case a mistake was made in the entry (someone else's number was accidentally added).
4.1.2.4.5 If UARS does not find a match,
4.1.2.4.5.1 UARS shall check that it is a well formed UPIN or Oscar Number
4.1.2.4.5.1.1 If the number is not well formed, UARS shall ask the provider to re-enter the number.
4.1.2.4.5.2 If the number is well formed, UARS shall inform the enrollee that they are not in the database, and to please provide the following information in addition to the UPIN information…
4.1.2.4.6 If UARS found a match, UARS displays the fields listed in the requirements at 4.1.1.. Otherwise, UARS asks the enrollee to fill in this information.
4.1.2.4.7 UARS shall ask for the enrollee's EIN, and allow them to re-enter if it is malformed.
4.1.2.4.8 UARS checks to see if the provider is an eligible Medicare Provider (a hospital, physician, physician group, ambulance service, Indian Health Service or tribal organization via the UARS repository of providers if a match is found. Else, from the enrollee entered information if no match was found in the repository).
4.1.2.4.8.1 If the provider is not eligible for UARS, UARS informs the provider of the ineligibility and ends the enrollment.
4.1.2.4.9 If the enrollee has a UPIN number (i.e. physician), UARS shall
4.1.2.4.9.1 Allow the physician to ask to list hospitals by state
4.1.2.4.9.2 Allow the physician to select hospitals where he/she will be submitting claims
4.1.2.4.9.3 Allow the physician to select hospitals from multiple states
4.1.2.4.10 If the enrollee has an Oscar number (i.e. hospital), UARS shall
4.1.2.4.10.1 Ask the enrollee if they will also be submitting charges for the on-call physicians. Yes/no field
4.1.2.4.10.2 If the answer is Yes, UARS shall ask the provider to provide a list of physician Medicare numbers for which they will be submitting charges.
4.1.2.4.11 If the provider lists a physician/hospital combination already enrolled in UARS, UARS shall not allow the enrollment of the provider to occur until the list contains only combinations not already enrolled with UARS. UARS shall allow the provider to edit their list and continue the enrollment.

4.1.2.4.12 UARS or the enrollee must then fill in the following (some of the following information must always be provided by the enrollee since the UPIN/Oscar information doesn't have it):
- 4.1.2.4.12.1 Provider Name (may be optional if a Physician Group or Hospital)
- 4.1.2.4.12.2 Provider Group Name (may be optional if this is an individual enrollee)
- 4.1.2.4.12.3 Provider Address (Street, City, Zip Code)
- 4.1.2.4.12.4 Provider Point-of-Contact/ Authorized Individual Name
- 4.1.2.4.12.5 Provider Point-of-Contact/ Authorized Individual Email
- 4.1.2.4.12.6 Provider Point-of-Contact/ Authorized Individual Phone Number
- 4.1.2.4.12.7 Password (entered twice)
- 4.1.2.4.12.8 State(s) of Service; UARS shall allow >=1 State
- 4.1.2.4.12.9 Current Medical Fiscal Intermediary or Carrier
- 4.1.2.4.12.10 Applicant's Tax ID Number
- 4.1.2.4.12.11 Applicant's Employer Identification Number
- 4.1.2.4.12.12 Payment Information
  - 4.1.2.4.12.12.1 Routing Number
  - 4.1.2.4.12.12.2 Bank Account Number
- 4.1.2.4.12.13 Authorized Representative Signature

4.1.2.4.13 Provider Hits the Submit Button

4.1.2.4.14 UARS shows the provider the information and asks for validation that the information is correct

4.1.2.4.15 If the provider responds that the information is not correct, UARS allows the enrollee to make corrections and submit again.

4.1.2.4.16 If the provider responds that the information is correct, UARS checks to see if the enrollee is in the UPIN or Oscar repositories. UARS shall check the enrollee EIN against the Oscar/UPIN EIN.
- 4.1.2.4.16.1 If they do not match, UARS shall notify the enrollee of the enrollment failure and contact the UARS administrator with this information
- 4.1.2.4.16.2 If they do match, UARS stores the new enrollee information and allows for the submittal of claims for this enrollee.

4.1.3 Alternate Scenario: Provider Not Eligible

In this scenario, they provide a valid Medicare number, but they are not an eligible provider. For example, a Skilled Nursing Facility.

4.1.3.1 Trigger

Medicare Provider wishes to Enroll with UARS

4.1.3.2 Precondition

Provider has a Medicare Provider Number

4.1.3.3 Expected Result

Provider is not Enrolled with UARS

4.1.3.4 **Provider ID returns a provider that is not eligible to UARS for claims**

4.1.3.4.1 Covered in Main Scenario. (See 4.1.3.3.8)

4.1.4 Alternate Scenario: Provider Enters Malformed Medicare Number

In this scenario, UARS is able to recognize the number is invalid

4.1.4.1 Trigger

Medicare Provider wishes to Enroll with UARS

4.1.4.2 Precondition

Provider has a Medicare Provider Number

4.1.4.3 Expected Result

Provider is not Enrolled with UARS

4.1.4.4 The provider is not in UARS's repository, and enters a number that is not in the appropriate format

4.1.4.4.1 UARS shall immediately inform the provider that the number is invalid, and allow them to enter a valid number. Covered in main scenario. (See 4.1.3.3.5.1.1)

B.3 Blank Template

1 Introduction

1.1 Document’s Intent

1.2 Definitions

1.3 Functional Purpose

The functional purpose describes *what* the system shall do.

1.4 Business Purpose

The business purpose describes *why* CMS would include funding in their budget for the system. The business purpose is always related to the mission of the organization and/or financial concerns.

1.5 Measures of Success

The measures of success list how the project team shall be measured against the business and functional purposes. The measures of success must always be concrete and measurable.

1.6 Stakeholders

Stakeholders of the system fall into four categories:

- Those who will pay for the system
- Those who will use the system
- Those who will gain an advantage by the system’s implementation
- Those who will lose an advantage by the system’s implementation

It is important to list and discuss all categories to accurately determine each group’s concerns and needs.

1.7 Risks & Assumptions

1.8 Mission Profile

There is always an inherent conflict between scope, budget available, schedule, and allowable defects. The mission profile helps the team determine what is most important, should a choice need to be made. The team is only allowed to make one check in the “High” importance column, so that everyone agrees on what is the highest priority.

Product Quality Dimension	Priority Level		
	Excel (High)	Improve (Medium)	Accept (Low)
Scope (features)			
Schedule			
Defects			
Resources (manpower, budget)			

2 Overview

2.1 Context

Below is a key to the types of symbols used in a context diagram.

□	Domains of Interest are subject matter areas, or activities, that have an effect on the business you are analyzing. May have slightly rounded corners or shaded to indicate different relationships
○	The Work Context, this is the business that is to be analyzed
← → ↔	Joins a Domain of Interest to the Work Context and why it is relevant to your study. This also shows the primary flow of data.

2.2 Work Context Diagram

The figure below shows the work context diagram for the system. The work context diagram shows all entities that will have knowledge of the system and that will interact with it. The direction of the arrows indicates which entity will *initiate* the event. After an event is initiated, there is usually two-way communication. The Work Context Diagram's arrows simply show who begins the events.

3 Business Level Requirements

3.1 Business Requirements

4 Domain to System Events and Requirements:

4.1 Event Name

4.1.1 Requirements

4.1.2 Scenario 1

This is the primary scenario

4.1.2.1 Trigger

4.1.2.2 Precondition

4.1.2.3 Expected Result

4.1.2.4 Steps

4.1.3 Alternate Scenario

4.1.3.1 Trigger

4.1.3.2 Precondition

4.1.3.3 Expected Result

4.1.3.4 Steps

Appendix C – Advanced Stylistic Approaches

Natural language is widely used in industry to state requirements for all types of systems, because it is flexible and universal. However, natural language has one major drawback, which is its inherent ambiguity. This appendix describes methods that may be used to reduce ambiguity in written requirements document.

C.1 Deletion

The process of deletion reduces the perception of a person to a scope they can deal with. In other words, things that seem to be not important are not recognized by a person or in the context of requirements not mentioned in a requirements document.

C.1.1 Implicit Assumptions

Implicit assumptions are those statements that are obvious or banal to the writer and are therefore omitted. Employ these techniques to aid in making implicit assumptions explicit.

Determine the main verb of a sentence. Form a new sentence by negation of the verb. Question: Which statements must be true in order to guarantee that both sentences make sense? All statements found may indicate an implicit assumption.

Requirement example:

> The Medicare Managed Care System shall assign a disenrollment date for a beneficiary in a Managed Care Organization when a termination date for the beneficiary's Part B eligibility is received.

The negation of this requirement would be:

> The Medicare Managed Care System shall not assign a disenrollment date for a beneficiary in a Managed Care Organization when a termination date for the beneficiary's Part B eligibility is not received.

The statements (assumptions) that must be true in order to guarantee that both sentences make sense include:

- that the beneficiary's Part B eligibility will be transmitted
- that the beneficiary is part of a Managed Care Organization

Under specified process words are verbs, adjectives and adverbs that aren't sufficiently quantifiable.

To detect under-specified process words determine the verbs of the sentence. If it is possible to have sentences with multiple actors something was deleted in the original sentence (e.g. the word transmit implies the questions what is transmitted, what are the aim and the source of the transmission).

C.1.2 Process Words

Process words are verbs, adjectives and adverbs. These types of words are often ambiguous and should be replaced by quantifiable measures.

To detect under-specified process words determine the verbs of the sentence. If it is possible to have sentences with multiple actors something was deleted in the original sentence (e.g. the word transmit implies the questions what is transmitted, what are the aim and the source of the transmission). For example:

- The system shall supervise the resources assigned to the patient care to ensure the maximum limit is not reached.

Use questions about the process words to determine if they are sufficiently specified:

- Supervise: what exactly is the system supervising and how?
- Resources Assigned: who or what can be assigned?
- Ensure: when and how is the maximum limit ensured?
- Maximum: What is the maximum limit?

C.1.3 Incomplete Superlatives and Comparisons

Incomplete superlatives and comparisons need a reference point to be completely specified. Moreover, there must be a meter and a predefined scale to compare certain aspects.

Use the following questions to detect incomplete superlatives and comparisons: Is there a reference point? What is the meter to measure a certain characteristic? What is the scale of the meter? For example:

- The functions should be easily changeable.

‘Easily changeable’ infers a reference point to something that is not easily changeable. What is the feature being compared to? Of course, ‘easily’ is also a process word that should be examined (see above).

C.1.4 Modal Operators of Possibility

In addition to the “what” a system shall do, the means to realize the “what” must be specified. Examples of modal operators of possibility include: *may, can* etc. Statements that describe a possibility or impossibility shall be analyzed by using the question: Which means enable or prevent a certain system reaction. For example:

- Updating a patient’s pre-existing conditions may not overwrite any existing history.

By what means is the update prevented?

C.1.5 Modal Operators of Necessity

Ensure that statements regarding the system's behavior contain the desired behavior and the exception behavior. Examples for modal operators of necessity include: must, shall, and should. Each sentence that contains a modal operator of necessity shall be inspected if it is necessary to define the behavior of the system in the case of an exception. For example:

- The system shall forward confirmation of the activity to the user.

What if forwarding services are not available or an invalid forwarding destination is provided?

C.2 Generalization

Generalization is defined as a process that leads to a detachment of an experience from its context and to assume that the experience is overall valid. In the requirements engineering process, generalization may lead to the omission of several behavior aspects of the system (e.g. special cases or error exceptions).

C.2.1 Universal Quantifiers

These elements are statements about a certain amount. They summarize a set of objects. Using universal quantifiers may lead to the problem that some objects summarized in the set do not realize the specified behavior. Universal quantifiers include: *never, ever, not any, everyone, no one, some one, nothing.*

Find all universal quantifiers. Check if the specified behavior of the system is really valid for all objects of the set summarized by the universal quantifier. Furthermore, check each sentence if there is an explicitly defined set of objects, the specified behavior is valid for. For example:

- Each communication shall be marked with a timestamp.

Every communication? Are there no exceptions?

C.2.2 Incomplete Specified Conditions

Conditions have the following sentence structure: *if condition then behavior1 else behavior2*. In requirements specifications the second case is often omitted. Key words to look for are: *when, then, if, in the case of, dependent on.*

Find the condition of the statement. Be sure that the requirement specifies the behavior for all possible cases (then and the else branch respectively). Additionally the engineer may check the following questions: Are all the possible conditions covered? Are all possible variants described? For example:

- In the case that the automatic data communication is disrupted, it should be possible to edit the individual fields of the transmission received.

What is possible if no disruption occurs?

C.2.3 Nouns without a Reference

This omission occurs when a noun of a statement is not specified sufficiently. Examples for insufficiently specified nouns are: *the user, the controller, the system, the message, the data, and the function*.

Check each noun of the requirements specification to determine if it specifies a defined person, a defined group of persons, or a defined real world object (group of objects). Insufficiently specified nouns can be identified by means of the following questions: *Who exactly? What in detail? Which part of the set?* Each noun specified insufficiently must be extended by a completion, which specifies the noun exactly. For example:

- The system shall display the data to the user electronically.

Which data exactly? Which users exactly?

C.3 Distortion

This phenomenon is related to the so-called nominalization, i.e., a noun stands for a complex process. Nominalization does not cause any problems as long as the process is unique, exactly specified and unambiguous in the current context. Examples of nominalizations include: the recording, the playback, the take off etc.

Check each noun of all sentences, with regard to the question: Is it possible to find a verb which describes a process and which is similar to the noun (for example, a ‘recording’ is both a noun and a verb)? You can use the following phrases:

- Is it possible to add the substantive to the phrase: a continuous...?
- Does the noun describe something that is not touchable?

If a question is answered with ‘yes’, the engineer has to check to see if any information was lost when describing a process by means of a noun. For example:

1. The system shall report on the status of the activity.

What is reported? Where is it reported?

C.4 Other Rules

An overall quality aspect of requirements statements is non-ambiguity. In the English language the frequent use of the gerund leads to misunderstandings and ambiguities. When using the gerund, the relation of the verb and the subject, performing the verb, is lost. For example:

- Reporting violations should be immediate.

This has two interpretations:

1. Violations should be reported immediately.
2. Reports of violations should appear immediately.

Appendix D – Common Working File Redesign Formatting

This appendix provides guidelines for creating and presenting quality business statements, business rules, business requirements and system requirements for the Common Working File Redesign (CWFR) project. This information is provided for any Requirements Engineer who is involved with CWFR related projects or who needs to write syntactically consistent requirements. Requirements written in this form are more readily designed and tested. The source of the information provided in this appendix is the Technical Writer’s Guide (TWG) used by the CWFR project.

CWFR is very different from typical projects because it uses a reverse re-engineering methodology where requirements are derived from the actual application code.

D.1 General Formatting

This section provides the general formatting instruction for business requirements, business statements, business rules and system requirements for topics in which the subject matter is common to all three areas. This includes:

- Government Printing Office (GPO) Style Manual
- General rules
- Capitalization
- Date format

D.1.1 GPO Style Manual

CMS uses the Government Printing Office (GPO) Style Manual as the basis for writing. The GPO Style Manual is available through the following URL: http://www.gpoaccess.gov/stylemanual/browse.html. In some cases, the TWG procedures depart from GPO rules. The TWG takes precedence over conflicting GPO rules.

D.1.2 General Rules

The following are general formatting rules for this TWG:

- Do not use quotes (") for dollar amounts, years, or edit numbers
- Use the $ for dollar amounts
- Zero dollars should be referred to as $0
- A single condition is placed on a single line, but multiple conditions must be on a separate line with a dash or multiple dashes as appropriate to the level
- Use for "ANY of the following", instead of "ONE of the following" for a list of elements.
- Use "ONE of the following is true" for instances when one of two or more conditions must be true

- Use of "One in the range of"
    - The revenue code is one in the range of "042x-044x" (therapies) or "055x-059x" (home health service or visits)
- Logic should be arranged using the following sequence when possible:
    - Transaction
    - Transaction line item
    - Stored claim
    - Stored claim line item
    - Using this sequence present single dashes conditions prior to double dash conditions
- Use the following construct for the UNLESS statement if they are all for the same type:

  UNLESS the transaction is:
  - a cancel, OR
  - for a home health agency payment that significantly exceeds total charges.

D.1.3 Capitalization

This TWG uses standard GPO capitalization rules. In general these include:

- Capitalize the first word in a sentence
- For data element names, capitalize the first letter of each word
- Acronyms, when approved for use, are in all capital letters (upper case in MS Word), e.g., DMERC, CWF
- Always capitalize Part A and Part B
- All other situations are in lower case

The exceptions to the standard GPO capitalization rules are:

- Use "AND" to connect multiple conditions that must all exist
- Use "OR" to connect multiple transactions that may exist, connect multiple bypass conditions, or connect conditions that may exist
- Use "WHEN" on the same line if there is a single condition. Use "WHEN" on a separate line with a colon if there are multiple conditions
- Use "UNLESS" in a business rule (on a separate line with a colon after and a comma on the preceding line) for all bypass conditions related to the business rule
- Use "BUT" when the conditions indicate an opposite condition for a business rule, system requirement
- Use "WHERE" with a colon to indicate conditions found in a stored claim
- Use "NOT", "ANY" or "ALL" or other such capitalization to emphasize any situation in a business rule or system requirement that may need clarity

D.1.3.1 Capitalization of Business Rules

The following are three examples of the use and exceptions to capitalization standards in business rules:

Business Rule Example 1:

CWF shall reject a home health transaction to add to stored claim data WHEN there are no claims for the transaction benefit year.

Business Rule Example 2:

CWF shall reject a home health transaction for Part A services
WHEN:
- the subscriber is not entitled to Part A benefits, OR
- the services are not within a Part A entitlement period,
UNLESS:
- the transaction is an original or replacement debit for a denied admission.

Business Rule Example 3:

CWF shall reject a carrier transaction WHEN the CLIA number is required BUT invalid,
UNLESS:
- the transaction is:
- - a cancel, OR
- - denied, OR
- - add to stored claim data, OR
- the line item is denied.

D.1.3.2 Capitalization of System Requirements

The following are two examples of the use and exceptions to capitalization standards in system requirements. Note that these examples are **not** for the same conditions as the above examples of business rules.

System Requirement Example 1:

The CWF system shall bypass consistency edit code 2102 for a hospice transaction WHEN the [Action Code] is:
- "4" (cancel), OR
- "7" (to add to stored claim data only).

System Requirement Example 2:

The CWF system shall bypass utilization edit code 6805 for a carrier OR DMERC transaction WHEN the [Approved HCPCS Code] is equal to ANY of the following: "90724", "90657", "90658", "90659", "90660", "G0008", "G0017", "90732", "G0009", "Q0124", "90669".

See the discussion under "Data Elements" in each category for information about the differences in material other than capital letters.

D.1.4 Date Format and Use of Slashes

The date format is MM/DD/YYYY, with slashes. Do not use slashes with any reference other than dates. Use English terms "and" or "or" as appropriate, instead of the slash. For example:

- 06/25/2004 for date display
- "hospital or SNF claim" instead of "hospital/SNF claim"

D.2 Multiple Conditions or Values

This section provides instruction and examples for multiple conditions or values for business rules and system in which the subject matter is common to both areas.

There are three basic situations that might result in multiple conditions:

- Where either of condition A or condition B results in an action. In this case use the word "OR" in upper case. A comma should precede the "OR"
- When both conditions must exist to result in the action. In this case use "AND" in upper case. A comma should precede the "AND"
- Capitalize "AND" and "OR" when they are connecting conditional statements or transaction types. Do not capitalize "and" or "or" when used within a single statement.
- When there is a combination of "AND" and "OR" conditions, these conditions **must** be clearly defined. If possible place the "OR" conditions at the end of the statement logic

This approach is applicable to two or more conditions:

Business Rule Example 1:

CWF shall reject a DMERC transaction
WHEN:
- the item is a capped rental item, electric wheelchair or a PEN pump, AND
- - the purchase option has been previously executed, OR
- - the maximum number of rental payments have been previously made,
UNLESS:
- the line item is denied, OR
- the service is for maintenance or servicing of the equipment, OR
- the transaction is a cancel.

This example shows that the presence of any one of the last two double dash conditions triggers the action.

Business Rule Example 2:

CWF shall reject a carrier OR DMERC transaction
WHEN:
- the line item indicates a dollar amount is applied to the Part B deductible, AND
- the same line item indicates the service is not subject to Part B deductible.

This example shows that all conditions must be present to trigger the action.

System Requirement Example 1:

The CWF system shall set utilization edit code D911 for a DMERC transaction
WHEN:
- the [Approved HCPCS Code] has a [HCPCS Payment Category Code] equal to "10" (PEN Pump), AND
- a [HCPCS Modifier Code] is NOT equal to:
- - "BR" (subscriber elected to rent DME equipment), OR
- - "BU" (subscriber did not reply to the DME purchase option), OR
- - "BP" (subscriber elected to purchase DME equipment), AND
- the stored [CMN Purchase Decision Indicator] is missing, AND
- the stored [CMN Total Services] is "13" (13 rental months).

This example shows multiple "AND" and "OR" conditions and nesting to clarify the conditions. The use of double dashes clearly indicates the "OR" conditions from the "AND" conditions.

This is inconsistent with CMS style guides used for Medicare manuals and all other CMS correspondence, which call for commas or semicolons without the "AND" or "OR" until the next to last line.

System Requirement Example 2:

The CWF system shall set consistency edit code 6803 for an outpatient, home health OR hospice transaction
WHEN:
- the [Value Code] is not ANY of the following: "12", "13", "14", "15", "41", "43", AND
- the subscriber has a [Primary Insurer] on the MSP stored data, AND
-stored MSP data is found for the subscriber WHERE:
-- the stored [MSP Validity Indicator] equals "y" (valid) or "I" (investigational MSP), AND
- - ONE of the following is true:
- - - the incoming [Statement From Date] and [Statement Through Date] are within the stored [MSP Effective Date] and [MSP Termination Date], OR

- - - the incoming [Statement From Date] and [Statement Through Date] overlap either the incoming [Statement From Date] and [Statement Through Date].

This example shows the use of "and" not capitalized when used within a single statement.

D.3 Punctuation, Spelling, and Usage

This section provides guidelines on punctuation, spelling, and usage for business rules and system in which the subject matter is common to both areas. This includes:

- Punctuation
- Spelling
- Usage

D.3.1 Punctuation

The following punctuation standards are to be used for all business rules and system requirements:

- All business rules and system requirements must end with a period.
- Make sure there is a comma before the "UNLESS"; then "UNLESS" is on a separate line and has a colon
- "WHEN" must have a colon if multiple conditions exist in the "WHEN" statement
- A comma must precede the "OR" in a list of "OR" conditions
- No commas will be used for the items in a list after the phrase "ANY of the following" and "ALL of the following." No "OR" will be used in the list as well. If there is no list but a row of items, then commas will separate the items, but not "or"

D.3.2 Spelling

The following spelling standards are to be used for all business rules and system requirements:

- Run MS Word and MS Excel spell-check feature prior to submitting any files for peer review and deliverable readiness review
- Use "through" instead of "thru"
- Use "crosswalk" instead of "cross-walk"
- Use "non payment" and "non covered" instead of "non-payment" and "non-covered"

D.3.3 Usage

The following verbs are to be used for all business rules and system requirements.

D.3.3.1 Verb Usage

Acceptable verbs for business rules and system requirements can be any of **receive, reject, approve, store, update, prepare, send (response or report) or other verb appropriate to the business rule or system requirement**. The edit processes will always reject or approve, and on approvals will store and perhaps may update, as the business rule and system requirement state. Another business rule or system requirement will send a record to an appropriate interface (such as the CWFMQA). Unsolicited response may occur on approvals or on rejects, depending upon the situation.

D.3.3.2 Letter "O" and the Number "0"

Use the letter key for the letter "O," and a "0" number key for the number "0." If the letter "O" is being represented, the parenthetical "(alpha)" must be used for clarity.

D.4 Terms and Requirements Hierarchy

One of CMS' largest system redesigns, the CWFR, is currently underway. The CWF interfaces with many other CMS-owned systems, as well as with a myriad of our business partners' systems. For this reason, it is felt that the guide user may need some familiarity with the requirements terms and requirements hierarchy of the CWFR project to understand how its nomenclature differs from this manual's.

D.4.1 Business Requirements

Business requirements are the high level features of the application to be developed. They are based on the business goals to be accomplished by the application. These business goals address legislative mandates or strategic objectives and include the core functionality of the system. Each business requirement (e.g., CWF shall validate all claims before payment is made.) is supported by multiple lower level business statements, business rules and system requirements.

D.4.2 Business Statements

Business statements define *what* the system should do (intent), and represent the analysis of multiple business rules to determine the higher level representation of the intent of the analyzed group of business rules. The business statements eliminate any of the implementation related components that may be inherent in the business rules. The business statements are also directly related to the business processes as modeled in the Business Process Model. Designing to the business statement level allows the system architects to explore multiple design choices with the minimum number of constraints necessary to realize a total solution with the best possible implementation.

D.4.3 Business Rules

Business rules are business or regulatory driven parameters that, if implemented in CWF, the system MUST programmatically support. The business rules embody the formal expression of a business, legal or regulatory request in English language. The business rule is written at the atomic level to represent a single business rule representing a single function. Since the system is complex, however, these business rules may be at varying levels of granularity. In this project, the code base is being used to derive the system requirements, and from them the business rules implemented in CWF. Each business rule must be associated with at least one or more system requirements.

D.4.4 System Requirements

System requirements define what the system does in order to cause the business rule to be implemented in CWF. Functional system requirements are specifically related to the purpose of the application and must be associated with a business rule. Non-functional system requirements are related to the operations, security or recovery of the application itself.

Non-functional systems requirements must also be associated with a business rule. Both types of system requirements are written in a vendor-neutral and implementation-neutral manner. An initial set of existing and proposed new system requirements are created based on the System Requirements Document (SRD) dated 12/13/2002.

D.4.5 Hierarchical Relationship

Business requirements are the highest level of requirements.

- Business Requirements will be traceable downward to Business Statements. A many-to-many relationship will be supported and every valid Business Requirement will be supported by at least one Business Statement
- Business Statements will be traceable downward to Business Rules. Again, the relationship will be many-to-many, although Business Rules will generally have only one or two parent Business Statements (never less than one)
- Business Rules will be traceable downward to System Requirements with a one-to-many cardinality (one Business Rule as the parent to multiple System Requirements). Every Business Rule will be supported by at least one System Requirement
- System requirements have bi-directional mapping to business rules and map to the Logical Data Model (LDM) using entity and attribute references as appropriate

D.4.6 Mapping to Guide's Nomenclature

While an exact mapping between two the methodologies is not possible, the following approximation is close:

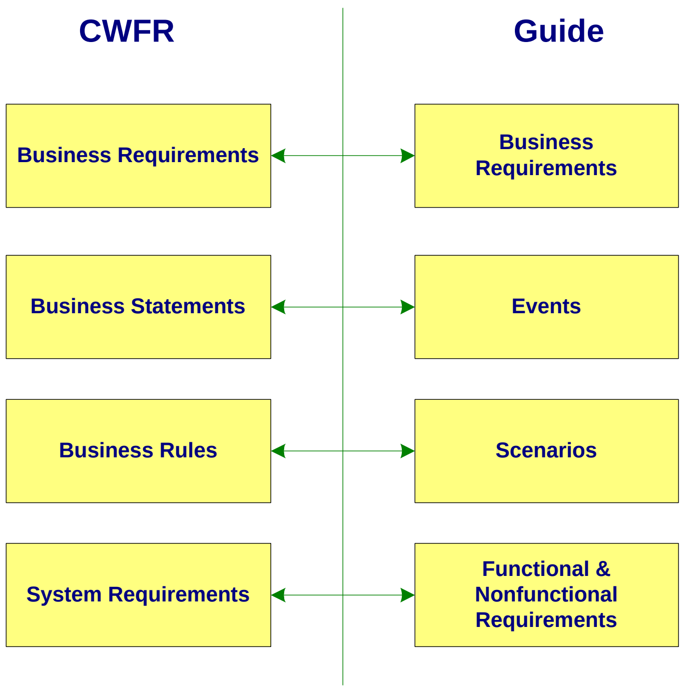


Figure 6 - Nomenclature Mapping

Index